1988-03

A computer program package for introductory one-dimensional digital signal processing applications

Hudik, Frank Edward

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

H8475

A COMPUTER PROGRAM PACKAGE FOR INTRODUCTORY ONE-DIMENSIONAL DIGITAL SIGNAL PROCESSING APPLICATIONS

by

Frank E. Hudik

March 1988

Thesis Advisor: D. E. Kirk

Y CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

ORT SECURITY CLASSIFICATION LASSIFIED	1b RESTRICTIVE MARKINGS
URITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.
LASSIFICATION / DOWNGRADING SCHEDULE R	
ORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)

IE OF PERFORMING ORGANIZATION POSTGRADUATE SCHOOL	6b OFFICE SYMBOL *(If applicable)* CODE 32	7a. NAME OF MONITORING ORGANIZATION NAVAL POSTGRADUATE SCHOOL
RESS *(City, State, and ZIP Code)* REY, CALIFORNIA 93943		7b. ADDRESS *(City, State, and ZIP Code)* MONTEREY, CALIFORNIA 93943
IE OF FUNDING / SPONSORING ANIZATION	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
RESS *(City, State, and ZIP Code)*		10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO

E *(Include Security Classification)*
PUTER PROGRAM PACKAGE FOR INTRODUCTORY ONE-DIMENSIONAL DIGITAL SIGNAL PROCESSING
CATIONS.

ONAL AUTHOR(S)
FRANK EDWARD

PE OF REPORT	13b TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15 PAGE COUNT
R'S THESIS	FROM ______ TO ______	1988, MARCH	294

PLEMENTARY NOTATION THE VIEWS EXPRESSED IN THIS THESIS ARE THOSE OF THE AUTHOR AND DO NOT
CT THE OFFICIAL POLICY OR POSITION OF THE DEPARTMENT OF DEFENSE OR THE U. S.
NMENT.

COSATI CODES			18 SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
LD	GROUP	SUB-GROUP	DIGITAL SIGNAL PROCESSING; DISCRETE SYSTEMS, COMPUTER ALGORITHMS; FREQUENCY RESPONSE; FOURIER TRANSFORM; DIFFERENCE EQUATIONS; SYSTEM STATES; CONVOLUTION; CORRELATION PERIODOGRAM

TRACT *(Continue on reverse if necessary and identify by block number)*

) EXISTED FOR A SET OF COMPUTER PROGRAMS WHICH COULD BE USED BY STUDENTS TO SOLVE
TARY DIGITAL SIGNAL PROCESSING PROBLEMS USING A PERSONAL COMPUTER. THIS PROJECT
VED THE DESIGN AND IMPLEMENTATION OF TEN ALGORITHMS THAT SOLVE SUCH PROBLEMS AND AN
IONAL ALGORITHM THAT CREATES PLOTS OF THE VARIOUS INPUT AND OUTPUT SEQUENCES. THE TWO
RY GOALS OF THE PROGRAMS WERE: 1) USER FRIENDLINESS AND, 2) PORTABILITY. WITH THESE
IN MIND, THE SOURCE CODE WAS WRITTEN USING FORTRAN-77 AND COMPILED BY A COMMERCIALLY
BLE FORTRAN COMPILER SPECIFICALLY DESIGNED FOR PERSONAL COMPUTERS. THE PLOTTING
M USES A FORTRAN-COMPATIBLE GRAPHICS PACKAGE THAT IS ALSO COMMERCIALLY AVAILABLE. THE
MS, ONCE COMPILED, CAN BE DISTRIBUTED TO USERS WITHOUT THE REQUIREMENT TO PURCHASE
A FORTRAN COMPILER OR A GRAPHICS PACKAGE HOWEVER, ACCESS TO A FORTRAN COMPILER
ES THE UTILITY OF THE PROGRAMS.

RIBUTION / AVAILABILITY OF ABSTRACT ICLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
ME OF RESPONSIBLE INDIVIDUAL STRUM	22b. TELEPHONE *(Include Area Code)* (408) 646-2652	22c. OFFICE SYMBOL CODE 62 St

RM 1473, 84 MAR 83 APR edition may be used until exhausted.
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

☆ U.S. Government Printing Office: 1986—606-24

A Computer Program Package for Introductory One-Dimensional Digital Signal Processing Applications

by

Frank Edward Hudik
Lieutenant, United States Navy
B.S., United States Naval Academy, 1979

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1988

ABSTRACT

A need exists for a set of computer programs that can be used by students to solve elementary digital signal processing problems using a personal computer. This project involved the design and implementation of ten algorithms that solve such problems and an additional algorithm that creates plots of the various input and output sequences. The two primary goals of the programs were: 1) user friendliness and, 2) portability. With these goals in mind, the source code was written using Fortran-77 and compiled by a commercially available Fortran compiler specifically designed for personal computers. The plotting program uses a Fortran-compatible graphics package that is also commercially available. The programs, once compiled, can be distributed to users without the requirement to purchase either a Fortran compiler or a graphics package; however, access to a Fortran compiler enhances the utility of the programs.

DISCLAIMER

TABLE OF CONTENTS

ACKNOWLEDGEMENT

Without the sacrifices of my wife and children I would not have been able to complete this project. I would like to thank my family for their sacrifices throughout the entire duration of the project. I would also like to thank my advisors Professor D. E. Kirk and Professor R. D. Strum whose foresight and experience prompted this project and whose encouragement guided this project to completion.

I. INTRODUCTION

Introductory digital signal processing courses present a wide spectrum of challenging topics to students enrolled in the electrical engineering curriculum. Undergraduate level courses present the transition from analog, continuous-time systems to digital, discrete-time processes. Topics include difference equations, state-matrix equations, system transfer functions, frequency response and the z-transform. Furthermore, the relationship between the time domain and the frequency domain is introduced and discrete convolution is discussed. Graduate level courses introduce filter design techniques and track the development of the Fast Fourier Transform (FFT) from the Discrete Fourier Transform (DFT). The interrelationships between circular convolution/correlation and the DFT are discussed and a brief introduction to spectral estimation is given.

In brief, the transition from the continuous-time method of system analysis to discrete-time methods is challenging to most students. While analytical methods are covered thoroughly in the classroom, computer solutions to problems reinforce learning by facilitating solutions to problems containing long sequences of data. Furthermore, the digital computer is the heart of digital signal processing applications in the real world. To this end, therefore, it is instructive to present computer programs that can perform many of the computations required for elementary digital signal processing (DSP).

The project summarized by this report involved designing a set of computer programs that can be used in a laboratory environment to reinforce the basic concepts of digital signal processing. Although deviations exist, the algorithms developed in First Principles of Discrete Systems and Digital Signal Processing, by R.D. Strum and D.E Kirk [Ref. 1] were used extensively in the development of these programs. The programs were written using Microsoft Fortran 77 Version 4.01.[1] This compiler was chosen because of its flexibility. It will compile Fortran programs for personal computers enhanced with a math coprocessor (8087/80287) or for less capable machines. The total project consists of ten programs related to the solution of digital signal processing problems and an additional program which produces 2-dimensional plots of the data. The plotting program was written using the Fortran-compatible Graphmatics software library.[2] The minimum hardware/software requirements necessary to run these programs are:

---

[1]Microsoft Corp., 1987, Bellevue, Wa. The programs will also compile after minor changes using Ryan McFarland Fortran Version 1.0 or later.

[2]Microcompatibles Corp., 1983, Silver Springs, MD.

* A personal computer with at least 320k of available memory.

* A monitor with a CGA card installed.[3]

* A single double-sided, double-density diskette drive.

The following options will either enhance the flexibility or increase the performance of the programs:

* A Microsoft Fortran Compiler Version 4.01 or later.

* A dot-matrix printer.

Chapter II details the scope of the programs and the general methodology used in developing them. Chapter III provides the concise development of each program. Flow-charts are presented to provide the architecture of the algorithms, depicting their macro-level design. Applicable equations are listed for each computational task and the corresponding Fortran implementation is discussed.

---

[3]A computer graphics card is only required to support the graphics program PLOTDAT.FOR and is not required for the other programs.

II. PROGRAM DEVELOPMENT

This chapter presents the scope of the computer programs and the general methods used in designing them. The goals of computational efficiency and user friendliness are addressed as the two sometimes conflict. Finally, a generic program structure used throughout the software development is presented.

A. SCOPE OF THE PROGRAMS

The package consists of ten problem solving programs and a two-dimensional plotting program. Each of the ten problem solving programs is oriented toward solving a specific DSP problem. The title of each program and its corresponding Fortran filename are listed below:

Problem Solving Programs

1. The frequency response of a digital filter.
   Filename: DIGFREQ.FOR

2. The frequency response of an analog filter.
   Filename: ANLGFREQ.FOR

3. The Discrete Fourier Transform (DFT) or Inverse DFT (IDFT) of a finite-length sequence.
   Filename: DFT.FOR

4. The periodogram of a finite-length sequence.
   Filename: PRDGRM.FOR

5. Convolution and correlation using the DFT algorithm.
   Filename: CONCORDT.FOR

6. The Fast Fourier Transform (FFT) or Inverse FFT (IFFT) of a finite-length sequence.
   Filename: FFT.FOR

7. Convolution and correlation using the FFT algorithm.
   Filename: CONCORFT.FOR

8. Convolution and correlation in the time domain.
   Filename: CONCOR.FOR

9. The iterative solution to a linear, time-invariant difference equation.
   Filename: DIFFEQ.FOR

10. The iterative solution to a set of linear, time-invariant state-matrix equations.
    Filename: STATEQ.FOR

Plotting Program

11. A file-driven, two-dimensional plotting algorithm.
    Filename: PLOTDAT.FOR

B. GENERAL METHODOLOGY OF PROGRAM DEVELOPMENT

All of the programs are oriented toward engineering students enrolled in elementary DSP courses. Since most engineering students have had at least some experience in Fortran programming, this language was an obvious choice. The programs can be executed without altering any of the Fortran code; however, some of the subroutines were specifically designed to allow the addition of Fortran statements to produce a desired sequence of data. This option will be discussed in more detail in Chapter III.

The source code structure of the algorithms is designed so that the user can follow the flow of each program as it performs the computations required by the task at hand. Computational efficiency is generally accepted to be one of a program designer's primary goals. However, because these programs were designed with the DSP student as the 'Target User', an overriding consideration was to make the flow of the programs understandable. For example, in CONCORDT.FOR the program will compute, among the options available, either the linear convolution or the circular correlation of two data sequences by using the DFT algorithm. The steps required by these two options are listed below [Ref. 1:pp. 424,432,433]:

<u>Option: Linear Convolution</u>

1. Zero pad array #1.
2. Zero pad array #2.
3. Compute the DFT of array #1.
4. Compute the DFT of array #2.
5. Multiply the results of steps 3 and 4.
6. Compute the IDFT of step 5.

<u>Option: Circular Correlation</u>

1. Compute the DFT of array #1.
2. Conjugate the result of step 1.
3. Compute the DFT of array #2.
4. Multiply the results of steps 2 and 3.
5. Compute the IDFT of step 4.

Clearly, if maximum efficiency was the only goal of the programs, the steps could be combined as follows:

<u>Option: Linear Convolution or Circular Correlation</u>

1. If option = Linear Convolution then zero pad array #1 and array #2.
2. Compute the DFT of array #1.
3. If option = Linear Correlation then conjugate the result of step 2.
4. Compute the DFT of array #2.
5. Multiply the results of steps 2 and 4.
6. Compute the IDFT of step 5.

However, by maintaining the separated algorithms, students can gain insight into the steps required to accomplish each of the tasks: linear convolution or circular correlation. This example, although somewhat contrived, demonstrates the general approach taken when confronted with the issue of efficiency versus readability throughout the programming. It is more instructive to separate the Fortran source code according to the steps required to perform a specific functional task rather than combine steps to form an efficient but less readable algorithm.

Each of the problem solving programs has two modes of operation: Test or Batch. Test Mode was conceived to guide inexperienced users through each program, allowing them the option of running the programs using data prestored in a data file named XXXX.TST. For example, while running the program DFT.FOR in Test Mode, the user can elect to use the prestored input data by entering 'DFT.TST', when prompted for the name of the input file. The prestored input data and the output which it produces correspond to an example problem developed in the header text of each program. The inexperienced user can therefore:

1) Read the header text including the sample problem.
2) Match the input parameters required by the program to those occurring in the input file: XXXX.TST.
3) Execute the program in Test Mode to produce the corresponding output. In Test Mode, key input parameters read from the input file are printed onto the monitor screen. This further aids inexperienced

users by providing the opportunity to detect invalid input.

The more experienced user can elect to run the programs in Batch Mode. In this mode the amount of interface with the user is minimized. Upon execution in Batch Mode, the program assumes that the appropriate input parameters have been stored in the default input file: XXXX.IN (e.g., DFT.IN). Figure 2.1 summarizes the events that occur in each of the two modes: Batch and Test.

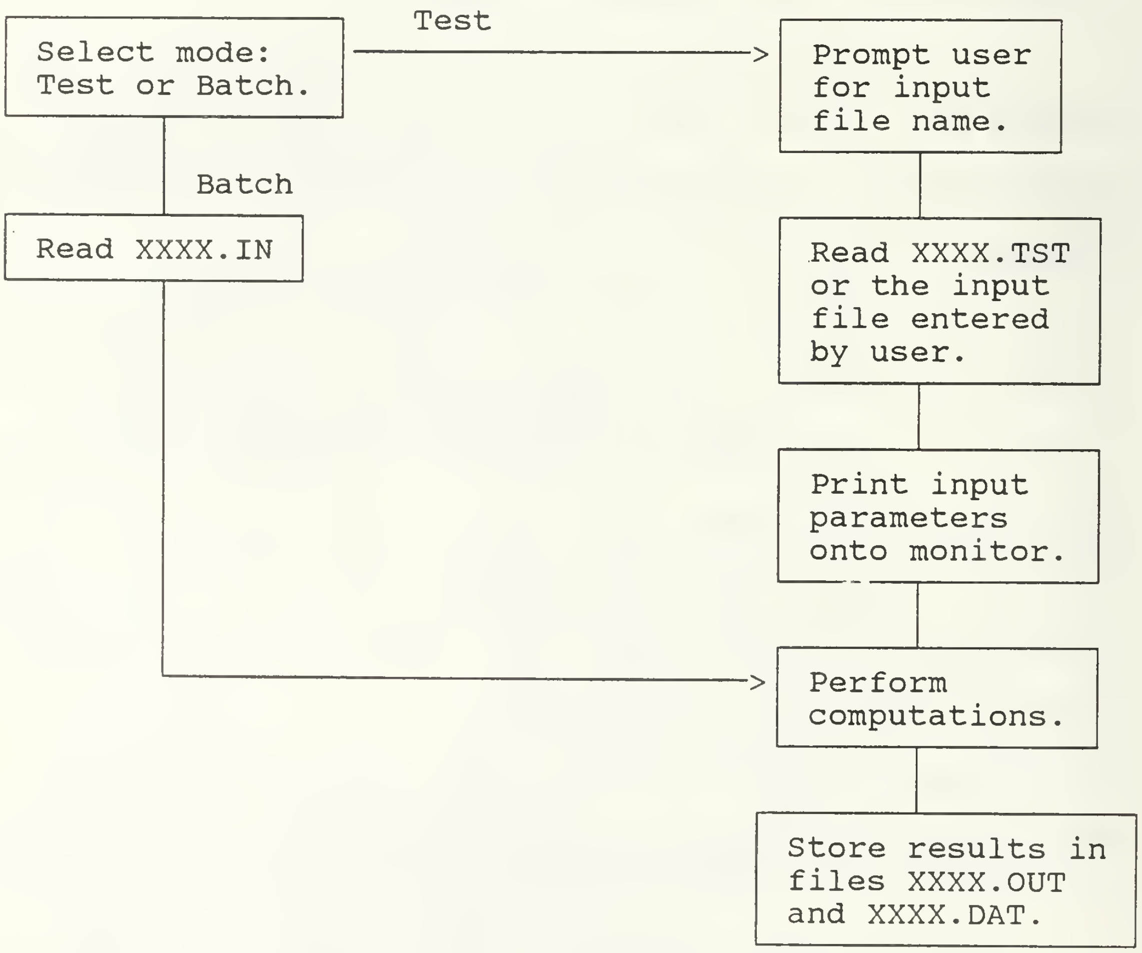

Select mode: Test or Batch.
Test
Batch
Read XXXX.IN
Prompt user for input file name.
Read XXXX.TST or the input file entered by user.
Print input parameters onto monitor.
Perform computations.
Store results in files XXXX.OUT and XXXX.DAT.

Figure 2.1 Program Flow.

C. GENERIC STRUCTURE

Students who use these programs will find some comfort in their standardized input and output structures as well as the documented Fortran code found in the algorithms themselves. With few exceptions, the specific names of variables correspond to their conceptual counterparts found in Reference 1. By using a single DSP textbook such as this to guide the choice of variable names, some standardization can be achieved throughout the programs' header text and accompanying source code. For example the following variable names, among others, occur in several of the programs: N = the number of output delays in a system; L = the number of input delays in a system; x() = the input sequence of a system; y() = the output sequence of a system, etc. A more comprehensive discussion of specific variable names occurs in the appendices. The remainder of this chapter is dedicated to describing the structure of the ten problem solving programs. The plotting program is not considered here as this program was designed with the sole purpose of reading data from an input file and creating a two-dimensional plot of the data.

1. Input Structure

All of the programs are file driven, that is, each program upon execution opens an input file, reads the contents of the input file, and performs the computations specified by the input parameters. Because of the variety of possible computations permitted by the programs, little attempt was made to standardize the specific inputs themselves. However, the following general specifications are used:

* All computations are performed using single precision arithmetic.
* All READ statements are format-directed; that is, none of the READ statements use list-directed format. An extensive discussion of the tradeoffs of these two methods is available in Fortran programming literature (e.g., [Ref. 2]).
* 'Real' numbers are read using format: F10.0. This allows the flexibility of reading real numbers entered using either F or E format descriptors.
* 'Complex' numbers are read as two real numbers, each having format: F10.0.
* Integer values required by the programs are read using the I (integer) format descriptor.
* Character strings required by the programs vary in length, however, none of the required string inputs is longer than 10 characters.
* Separate data entries occurring on a given line (record) of the input file begin in one of the following columns: 1, 11, 21, 31, 41, 51.

During the developmental stage of the programs an attempt was made to use list-directed inputs. It was discovered however, that different Fortran compilers treat variable assignments in different ways. For example, while some compilers allow integers to be read into variables declared as real and vice versa, other compilers will not allow this. In order to maintain the portability of the programs; therefore, format-directed inputs are used exclusively.

Each of the programs contains instructions in the header text describing the options available and the input parameters required to run the program. An example problem is developed in this text including a brief overview of the problem, the input required to achieve the desired results, and a listing of the actual output produced by the program. This approach allows first-time users to confirm their understanding of each program's input requirements and corresponding format. As stated at the beginning of this chapter, each sample problem can actually be run by executing the program in Test Mode and specifying the input filename: XXXX.TST at the prompt. Experienced users can elect to run the programs in Batch Mode in which case the programs attempt to OPEN and READ the default input file: XXXX.IN (e.g., DFT.IN). Input files for other than test runs should be named according to this scheme.

2. Program Structure

Each program consists of a main program and one or more subroutine subprograms. The computations related to the functional tasks of each program are performed in suitably named subroutines. For example, the program DFT.FOR will compute either the Discrete Fourier Transform or the Inverse Discrete Fourier Transform of a given input sequence depending on the option selected. The program consists of a main program and the subroutines DFT, INVDFT, and SAMPLE. The subroutines DFT and INVDFT perform the computations suggested by their names and SAMPLE allows the user the option of generating an input sequence by providing the appropriate Fortran statements in the space provided in the body of the subroutine source code. Housekeeping tasks are reserved for the main programs. These tasks include, but are not limited to, the following:

* Obtaining the inputs required by the program, either from the input file or from the keyboard, as appropriate.
* Conducting rudimentary error checks on the input data.
* Calling the appropriate subroutines to perform the desired computations.
* Performing data conversions (e.g., Real and Imaginary --> Magnitude and Phase).
* Creating the output files.

Error checking consists of ensuring that the numerical input values are within the range specified in each program's header text. This reduces the chance of making gross errors such as inputting '30' when a READ statement requires format I4 thereby producing an erroneous input of 3000! Character string inputs are used by most of the programs to distinguish among the available options. Error checking involving these inputs is limited to a simple string comparison. The error messages produced by any of these algorithms are self-explanatory.

3. Output Structure

Several purposes are served by the output listings of the programs. Among these are the following:

1) To allow the user to confirm anticipated results by comparing the output data generated by the computer algorithm to analytical results generated independently.

2) To place the output data in a format suitable for two-dimensional plotting by a program such as PLOTDAT.FOR.

The former stated purpose requires that the output data be in easily readable, tabular form. To accomplish this, each program generates an output file named: XXXX.OUT (e.g., FFT.OUT). At the beginning of each tabular output file the data read from the input file is listed including any input sequence(s). Additionally, any input sequence(s) generated by a subroutine such as SAMPLE is also written to the output file. Lastly, the output data generated by the program is listed. This comprehensive listing of the input data as well as the output data allows the user to verify that the input values were read correctly from the input file. Furthermore, with both the input data as well as the output data in one listing, the user can identify the problem and check the computational results more readily.

The two-dimensional plotting program PLOTDAT.FOR reads data according to the format: f12.0, 2X, f12.0, with the first entry on each line corresponding to the ordinate value and the second, the abscissa. The plotting program will produce more than one plot if the appropriate data entries exist in the input file. This flexibility of the plotting program suggests the possibility of plotting not only the output data, but also any input sequence(s). To accommodate the capabilities of the plotting program, each of the ten problem solving programs creates an output file named: XXXX.DAT (e.g., FFT.DAT). For programs that require an input sequence(s), both the input sequence(s) as well as the output sequence(s) are stored in the output file XXXX.DAT. These output files are created in addition to the tabular output files and do not require any user interface.

The general content and format of the programs have been discussed in this chapter. The next chapter formally develops each program, relating computational goals to specific source code.

III. SOFTWARE DESIGN

A subsection of this chapter is dedicated to each of the ten problem solving programs as well as the plotting program PLOTDAT.FOR. Flowcharts that describe the various algorithms are located in Appendices A through K and listings of the Fortran source code for the programs are included as Appendix L.

A. PROBLEM SOLVING PROGRAMS

1. DIGFREQ.FOR

The program DIGFREQ.FOR is designed to compute the frequency response of up to three digital filters. The program assumes that the filters are stable and that the transfer function of each filter has the form:

$$H(z) = \frac{b(0)z^{L} + b(1)z^{L-1} + b(2)z^{L-2} + \ldots + b(L-1)z + b(L)}{c(0)z^{N} + c(1)z^{N-1} + c(2)z^{N-2} + \ldots + c(N-1)z + c(N)} \tag{3.1}$$

The order of the numerator (L) and the order of the denominator (N) can be assigned any integer values in the range: 0 to 128. These parameters are read from the input file. The program accepts up to three distinct filter equations and will compute the magnitude and phase (degrees) for up to 101 frequency points for each filter. The frequency (Θ) range of interest is also specified by the user in the input file.

The program consists of the main program DIGFREQ.FOR and the subroutines COEFF and DFRESP. The user can provide the filter coefficients [arrays b() and c()] in the input file or can elect to generate them through use of the subroutine COEFF. If this latter option is chosen then the user must provide the appropriate Fortran statements in the space allocated in the subroutine and the program must be compiled again before execution. Subroutine DFRESP is called by the main program to perform the actual frequency response computations.

This program is an implementation of the psuedocode presented in Reference 1. [p. 203] The software flowcharts of Appendix A depict the overall program structure. If the user has elected to run the program in Batch Mode the default input file DIGFREQ.IN is opened by the program and the input parameters are read from it. If Test Mode is chosen, the input file whose name is specified by the user is read. The parameters describing each filter are passed to subroutine DFRESP which then computes the magnitude and phase of H(z) for each value of $z = e^{j\Theta}$ in the specified range of Θ. By using nested multiplication to compute the frequency response, DFRESP adds a measure of efficiency to the program [Ref. 3]. In the limit (L = 128, N = 128), if evaluation of each polynomial term is performed for 101 frequency points, the total number of complex multiplies required is over 1.6 million. The corresponding number of complex multiplies required using the Nested Multiplication technique is about 26 thousand.

The input parameters read from the input file and the corresponding frequency response(s) generated by the program are stored in tabular form in the output file DIGFREQ.OUT. Additionally, the program writes the frequency response data into the file DIGFREQ.DAT. Appropriate labels and control parameters accompany the data in DIGFREQ.DAT and are written in a form compatible with the plotting program PLOTDAT.FOR.

Appendix A traces the development of two digital filters and compares the anticipated frequency responses with the computer generated output. Plots of the output data produced by PLOTDAT.FOR are included in the analyses.

2. ANLGFREQ.FOR

The program ANLGFREQ.FOR is designed to compute the frequency response of continuous-time (analog) systems. The design of ANLGFREQ.FOR is very similar to DIGFREQ.FOR. The program assumes that the filter is stable and that the transfer function has the form:

$$H(s) = \frac{b(0)s^{L} + b(1)s^{L-1} + b(2)s^{L-2} + \ldots + b(L-1)s + b(L)}{a(0)s^{N} + a(1)s^{N-1} + a(2)s^{N-2} + \ldots + a(N-1)s + a(N)} \qquad (3.2)$$

The order of the numerator (L) and the order of the denominator (N) can be assigned any integer values in the range: 0 to 128. The parameters L and N, as well as the coefficients b(0), ..., b(L) and a(0), ..., a(N) are specified in the input file. As with DIGFREQ.FOR, the program will accept up to three distinct filter equations from the input file, and calculate the magnitude and phase for up to 101 frequency points for each filter. The user must specify the frequency range of interest and can elect to have the magnitude expressed in decibels (dB).

The algorithm consists of the main program ANLG-FREQ.FOR and the subroutine AFRESP. The main program controls the input and output and calls subroutine AFRESP to compute the frequency response for each filter. The software flowcharts of Appendix B depict the program structure. Subroutine AFRESP is an adaptation of the Fortran source code used in subroutine DFRESP [Ref. 1:p. 621]. ANLGFREQ.FOR computes the frequency response of filters expressed in the 's domain'. This differs from DIGFREQ.FOR which computes the frequency response of filters expressed in the 'z domain'. Notwithstanding this difference, the subroutines DFRESP and AFRESP are identical [except that 'jw' (j omega) is substituted for the complex variable 'z' in AFRESP] and the efficiencies gained through the use of nested multiplication apply for AFRESP as well.

The output data produced by ANLGFREQ.FOR are stored in two files: ANLGFREQ.OUT and ANLGFREQ.DAT. The former output file lists both the input parameters as well as the output data for each filter in tabular form. ANLGFREQ.DAT is a listing of the output data in a form suitable for plotting.

Appendix B presents the conceptual development of an analog filter and its corresponding frequency response using ANLGFREQ.FOR. Plots of the output data generated by PLOTDAT.FOR are included in the analysis.

3. DFT.FOR

A task particularly well suited for the digital computer is the computation of the Discrete Fourier Transform (DFT) or its inverse, the IDFT. The DFT of a sequence N samples long is defined by:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi nk/N}, \qquad k = 0, 1, \ldots, N-1 \tag{3.3}$$

Its corresponding inverse, the IDFT, is defined by:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)e^{j2\pi nk/N}, \qquad n = 0, 1, \ldots, N-1 \tag{3.4}$$

An alternate method for calculating the IDFT is the 'Alternate Inversion Formula' [Ref. 1:p. 406]:

$$x(n) = \frac{1}{N}\left[\sum_{k=0}^{N-1} X^{*}(k)\, e^{-j2\pi nk/N}\right]^{*}, \quad n = 0, 1, \ldots, N-1$$

* Denotes complex conjugation. (3.5)

Once an algorithm has been developed to compute the DFT [Equation (3.3)], the IDFT can be efficiently computed using Equation (3.5). The steps involved in computing the IDFT are summarized below.

1. Conjugate all N values of the sequence X(k).
2. Use the DFT algorithm to compute the DFT of the conjugated sequence.
3. Conjugate the sequence resulting from step 2 and divide each value by N. The result is the sequence x(n).

The program DFT.FOR consists of a main program and the subroutines DFT, INVDFT, and SAMPLE. The main program reads the required input parameters from the input file and, depending on the option specified by the user, computes either the DFT or the IDFT of the input sequence by calling the appropriate subroutine. The program will accept a complex input sequence of up to 256 samples. Since typing a long sequence of data into an input file is somewhat impractical, the user has the option of generating the input sequence by using the subroutine SAMPLE. If this method of data generation is chosen, the user must provide the Fortran statements required to generate the sequence, in the space provided in the subroutine. For example, if the user desires to compute the DFT of the 'real' sequence xin(i) = cos(iπ/N) for i = 0, 1, ..., N-1 appropriate Fortran statements to be written into SAMPLE are:

```
        do 100 i = 0, N-1
          xin(i) = cmplx(cos(i*3.14159/N), 0.0)
    100   continue
```

A caveat to using this method of data generation is that the program must be recompiled before execution.

The software flowcharts of Appendix C depict the overall program structure. As the flowcharts indicate, the program computes the DFT of the sequence according to Equation (3.3) and computes the IDFT by use of the Alternate Inversion Formula. Thus, subroutine INVDFT must call subroutine DFT as part of the IDFT computation. Both algorithms are implementations of the psuedocode architecture presented in Reference 1 [pp. 411, 412]. This design incorporates the efficiencies of nested multiplication stated previously. Also included in Appendix C are three example problems that demonstrate the capabilities of DFT.FOR.

4. PRDGRM.FOR

As the first of two application programs of the DFT algorithm, PRDGRM.FOR provides an introduction to the classical methods of spectrum analysis. The periodogram of a sequence is defined, quite simply, as the square of the DFT of the sequence divided by N, the number of points [Ref. 1:pp. 454-456]. Equation (3.3) governs the computation of the DFT sequence X(k), and with this task accomplished the periodogram can be computed from:

$$Sxx(k) = \frac{1}{N} X(k)\overset{*}{X}(k) \qquad k = 0, 1, \ldots, N-1$$

* Denotes complex conjugation. (3.6)

The program PRDGRM.FOR will compute the periodogram of a sequence consisting of up to 256 complex values. To facilitate the generation of long sequences of input data, the subroutine SAMPLE has been provided as part of the algorithm. Instructions for its use are the same as those discussed in the previous section. The main program performs the input and output tasks required by the program and calls subroutine DFT to compute the sequence X(k). Once the array X(k) has been computed, the main program computes the sequence Sxx(k) by implementation of Equation (3.6). An option available to the user is to have the output sequence expressed in decibels, a result commonly referred to as the 'Log Periodogram'.

Appendix D includes three example problems that demonstrate the utility of PRDGRM.FOR. Included in the problem analyses is a discussion of the limitations of the periodogram as a means of spectral estimation for finite-length sequences. The software flowcharts of the appendix outline the program's structure.

5. CONCORDT.FOR

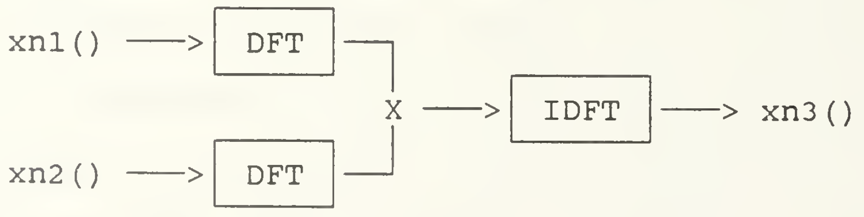
The second application program of the DFT algorithm is CONCORDT.FOR. The program uses the DFT technique to perform one of the following operations, given two sequences of complex input values:

1. Linear convolution.
2. Linear correlation.
3. Circular convolution.
4. Circular correlation.

The program consists of the main program CONCORDT.FOR and the subroutines DFT, INVDFT, SAMPL1, SAMPL2, and ZEROPAD. Subroutines DFT and INVDFT compute, respectively, the DFT and the IDFT of a given sequence. Subroutines SAMPL1 and SAMPL2 allow the user the option of generating the input sequences xn1() and xn2() by providing the appropriate Fortran statements in the space provided in the subroutines. Details of these four subroutines were presented previously and therefore will not be repeated here. Subroutine ZEROPAD is designed to extend each of the input sequences to the length required for computing either the linear convolution or the linear correlation using the DFT technique. For example, to compute the linear convolution of the two input sequences xn1() and xn2() of length N1 and N2, respectively, each of the these sequences must be padded with enough zeros to extend the sequences to length: N3 = N1 + N2 - 1 samples. To accomplish this, the sequence xn1() must be padded with N3 - N1 zeros and the sequence xn2() must be padded with N3 - N2 zeros. This same procedure is required if the linear correlation of the two sequences is to be performed. In either case, subroutine ZEROPAD extends the sequences to the required length. The program accepts input sequences consisting of up to 128 complex values. The input sequences xn1() and xn2() are assumed to exist in the sample intervals 0 to N1 - 1 and 0 to N2 - 1, respectively.

The technique of using the DFT algorithm to compute the convolution of two sequences is based on the concept that time domain convolution corresponds to frequency domain multiplication. Given this relationship, the steps required to compute the circular convolution of two sequences are depicted below. For ease of documentation, the following symbols are used:

$*$	Denotes linear convolution.
$\circledast$	Denotes circular convolution.



$$xn3() = xn1() \circledast xn2()$$

Figure 3.1 Circular Convolution.

The circular correlation $\tilde{R}_{xn1xn2}()$ can be computed in the time domain by reversing the order of the sequence xn1() and performing circular convolution on the resulting sequences. Circular correlation can also be performed using the DFT technique by performing the following steps:

$$\begin{tikzcd}
xn1() \arrow[r] & \boxed{\text{DFT}} \arrow[r] & \boxed{\text{CONJUGATE}} \arrow[dr, dash] & & & \\
& & & X \arrow[r] & \boxed{\text{IDFT}} \arrow[r] & \tilde{R}_{xn1xn2}() \\
xn2() \arrow[r] & \boxed{\text{DFT}} \arrow[urr, dash] & & & &
\end{tikzcd}$$

$\tilde{R}_{xn1xn2}()$ = The circular correlation of xn1() and xn2().

Figure 3.2 Circular Correlation.

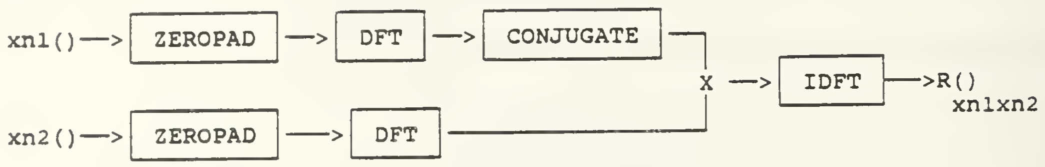
Linear convolution is performed by first zero padding the sequences to length N3 = N1 + N2 - 1 and then performing circular convolution on the extended sequences. Thus, the steps required to perform linear convolution are:

$$\begin{tikzcd}
xn1() \arrow[r] & \boxed{\text{ZEROPAD}} \arrow[r] & \boxed{\text{DFT}} \arrow[dr, dash] & & & \\
& & & X \arrow[r] & \boxed{\text{IDFT}} \arrow[r] & xn3() \\
xn2() \arrow[r] & \boxed{\text{ZEROPAD}} \arrow[r] & \boxed{\text{DFT}} \arrow[ur, dash] & & &
\end{tikzcd}$$

xn3() = xn1() * xn2()

Figure 3.3 Linear Convolution.

Linear correlation is similarly computed by first zero padding the sequences and then performing circular correlation on the extended sequences. The steps required for this computation are:



$R_{xn1xn2}()$ = The linear correlation of xn1() and xn2().

Figure 3.4 Linear Correlation.

A phenomenon encountered when performing the linear correlation operation, using the DFT technique, is 'wrap-around' of the output sequence. This is directly attributable to the required zero padding of the input sequences. While it would be a simple matter of software manipulation to prevent the wraparound from appearing in the final output sequence, it is felt that incorporation of the phenomenon is relevant to student comprehension of the actual computations involved. Example #4 of Appendix E demonstrates the wraparound that occurs when the linear correlation of two input sequences is computed.

The procedures described above as well as a comprehensive analysis on use of the DFT technique to perform convolution and correlation are presented in Chapter

7 of Reference 1. CONCORDT.FOR is a Fortran implementation of this technique. Flowcharts describing the structure of CONCORDT.FOR and subroutine ZEROPAD are included in Appendix E. Also included in this appendix are example problems demonstrating the four computations that CONCORDT.FOR is capable of performing.

6. FFT.FOR

Similar in purpose to DFT.FOR, FFT.FOR is designed to compute the DFT or the IDFT of a complex input sequence consisting of up to 256 samples. The advantage FFT.FOR has to offer over DFT.FOR is use of the Fast Fourier Transform (FFT) technique for computing the DFT. Entire books have been dedicated to this subject, most of which include a Fortran algorithm for performing the FFT computation. The FFT technique used by FFT.FOR is a Radix-2, Decimation In Time algorithm adapted from the psuedocode design appearing in Reference 1 [pp. 510-512]. Included in this reference is a software flowchart of the subroutine REVERSAL which is discussed in more detail below.

The program consists of the main program FFT.FOR and the subroutines FFT, REVERSAL, INVFFT, and SAMPLE. The main program calls subroutine FFT to perform the actual FFT computations. In order to use the Decimation In Time algorithm, the input sequence must be reordered according to a 'bit-reversal' scheme. This scheme involves changing the position that each sample holds in the input sequence by reversing the order of the bits corresponding to the positional address of each sample. For example, an 8-sample input sequence would have the binary positional addresses: [000 001 010 011 100 101 110 111]. After reversing the order of the sequence, the binary addresses of the bit-reversed sequence would be: [000 100 010 110 001 101 011 111]. In terms of the Fortran array x(), the 8-element, bit-reversed array will contain the original eight values but rearranged into the new order:

```
x(0)
x(4)
x(2)
x(6)
x(1)
x(5)
x(3)
x(7)
```

Subroutine REVERSAL performs this reordering of the input sequence. The main program passes the original input sequence to the subroutine in the array xtmp() and the subroutine returns the bit-reversed sequence in the array x().

In addition to the FFT computation, FFT.FOR was designed to compute the Inverse Fast Fourier Transform (IFFT). If used for no other reason, the IFFT serves as a check on the FFT computation. For example, the user can elect to have the FFT of a sequence computed, and as a check on the computed results, run the program again but this time using the FFT results as input to the IFFT computation. The results of this second run should be the original input sequence, with some allowance for single-precision roundoff error. Subroutine INVFFT performs the IFFT computation in a manner identical to the IDFT computation performed by subroutine INVDFT. In fact, the flowcharts of the two subroutines, except for the names of the variables, are identical.

Subroutine SAMPLE provides the means to generate the input sequence by allowing the user to write the appropriate Fortran statements into the space allocated in the subroutine. The user can elect to use this method of data generation or can choose to provide the N complex input samples in the input file.

Because FFT.FOR is a Radix-2 algorithm, the input sequence must be of length $N = 2^m$, m = integer. This apparent limitation to the utility of the FFT is easily overcome in practical applications by either: 1) Requiring the sampled input sequence to be of the correct length or; 2) Zero padding the input sequence until it is of length $N = 2^m$. This later technique is used in the program CONCORFT.FOR presented in the next section. Zero padding should not be used to extend a sequence for the purpose of computing a periodogram; however, since the addition of zeros will cause erroneous frequency information to appear in the periodogram sequence.

The input and output sequences are stored in tabular form in the file FFT.OUT. Additionally, the sequences are written into the file FFT.DAT in a form suitable for plotting.

The software flowcharts of Appendix F depict the structure of FFT.FOR and the subroutines FFT, INVFFT and REVERSAL. Also included in this appendix are two example problems that demonstrate both the FFT and the IFFT computations.

7. <u>CONCORFT.FOR</u>

As an application program for the FFT algorithm, CONCORFT.FOR is capable of performing any one of the following four operations, given two sequences of complex input data:

1. Linear convolution.
2. Linear correlation.
3. Circular convolution.
4. Circular correlation.

Similar in design to CONCORDT.FOR, this program also uses the DFT technique to perform the selected operation. Figures 3.1 through 3.4 describe the computations required by each of the four operations. In order to take full advantage of the efficiencies of the FFT algorithm, however, each of the DFT computations required by CONCORFT.FOR is accomplished using an FFT. This design invokes the requirement that the input sequences be of length $N = 2^m$ (m = integer), since the FFT subroutine used by the program is a Radix-2 algorithm. For the linear convolution/

correlation operations this requirement is easily fulfilled by zero padding the sequences to a suitable length. For example, if the sequences xn1() and xn2() are of length N1 = 4 and N2 = 3, the linear convolution/correlation operations, as computed via the DFT technique, require that the input sequences be zero padded to the minimum length: N3 = N1 + N2 - 1 = 6. Since N3 = 6 is not an integer power of two, CONCORFT.FOR will further extend the sequences to length N3 = 2^3 = 8 by additional zero padding. The extended sequences can then be used in the FFT computations.

While zero padding is intrinsic to CONCORFT.FOR's linear convolution/correlation operations, its use in performing circular convolution/correlation will lead to erroneous results. For this reason, CONCORFT.FOR will perform circular convolution/correlation only if the input sequences are of equal length and the lengths are an integer power of two. The program is designed to screen the input data to ensure that these requirements are met. Suitable error messages are printed on the screen and the program's execution is halted if they are not met.

CONCORFT.FOR accepts input sequences consisting of up to 128 complex values. The input sequences xn1() and xn2() are assumed to exist in the sample intervals 0 to N1 - 1 and 0 to N2 - 1, respectively. The main program controls the input/output tasks and calls the appropriate subroutines to perform the selected operation. There are six subroutines in all and a brief description of the function of each subroutine is as follows:

1. FFT - Computes the Fast Fourier Transform of a sequence.

2. INVFFT - Computes the Inverse Fast Fourier Transform of a sequence.

3. REVERSAL - Rearranges a sequence into bit-reversed order.

4. ZEROPAD - Extends a sequence by adding an appropriate number of zero values.

5. SAMPL1 - Allows the user the capability of generating the sequence xn1() by providing the appropriate Fortran statements in the space allocated in this subroutine.

6. SAMPL2 - Allows the user the capability of generating the sequence xn2() by providing the appropriate Fortran statements in the space allocated in this subroutine.

A software flowchart describing the design of CONCORFT.FOR is provided in Appendix G. Also included in this appendix are four example problems, each of which demonstrates one of the operations that CONCORFT.FOR is capable of performing.

8. CONCOR.FOR

The program CONCOR.FOR is designed to compute either the linear convolution or the linear correlation of the two input sequences xn1(n) and xn2(n). The non-zero values of the sequence xn1(n) must exist in the range: $ns1 \le n \le ne1$. Similarly, the non-zero values of xn2(n) must exist in the range $ns2 \le n \le ne2$. The constraints on the values ns1, ne1, ns2, ne2 are: $-128 \le ns1 \le ne1 \le 128$ and $-128 \le ns2 \le ne2 \le 128$.

Unlike the frequency domain techniques used to perform convolution and correlation in CONCORDT.FOR and CONCORFT.FOR, all computations performed by this algorithm are done in the time domain. For linear convolution, Equation (3.7) applies.

$$yn(n) = \sum_{m=-\infty}^{\infty} xn1(m)*xn2(n-m) \tag{3.7}$$

For linear correlation, as it is performed by this algorithm, Equation (3.8) applies.

$$R_{xn1xn2}(p) = \sum_{m=-\infty}^{\infty} xn1(m)*xn2(p+m) \tag{3.8}$$

The program consists of the main program CONCOR.FOR and the subroutines SAMPL1, SAMPL2, CONVOL, and CORREL. Subroutines SAMPL1 and SAMPL2 allow the user to generate either of the input sequences by providing the appropriate Fortran statements in the space provided in the subroutines. Subroutine CONVOL is called by the main program to compute the linear convolution of the two sequences, according to Equation (3.7). The computations are necessarily limited to include only the non-zero ranges of the two input sequences. Subroutine CORREL is called by the main program to compute the linear correlation of the two input sequences, according to Equation (3.8). Similar to CONVOL, the computation is limited to the non-zero ranges of the input sequences. An alternate method of computing the linear correlation would be to reverse the sequence of values stored in xn1(n) and then to compute the linear convolution of the resulting sequences [Ref. 1:p. 432]. This method is not used in this algorithm.

Appendix H contains flowcharts that describe the main program, as well as the subroutines CONVOL and CORREL. The appendix also includes example problems that demonstrate the performance of CONCOR.FOR.

9. <u>DIFFEQ.FOR</u>

The program DIFFEQ.FOR is designed to compute the iterative solution to a linear, time-invariant (LTI) difference equation. The program will compute the solution for up to four distinct equations, each of the form:

$$\begin{aligned} y(ns) = {} & a(1)*y(ns-1) + a(2)*y(ns-2) + \ldots + a(N)*y(ns-N) + \\ & b(0)*x(ns) + b(1)*x(ns-1) + \ldots + b(L)*x(ns-L) \end{aligned} \tag{3.9}$$

The solution to each equation is computed for values of ns in the range $0 \le ns \le nstop$, where nstop can be assigned any integer value in the range $0 \le nstop \le 300$. The input sequence x(ns) is assumed to be zero for values of ns less than zero. The parameter L corresponds to the maximum number of delays in the input sequence and can be assigned any integer value in the range $0 \le L \le 128$. Similarly, the parameter N corresponds to the maximum number of delays in the output sequence and can be assigned any integer value in the range $0 \le N \le 128$.

To run the program, the user must provide the parameters N, L and nstop, as well as the coefficients a(1)...a(N), and b(0)...b(L). For values of N > 0, the user must also provide the initial condition sequence y(-N)... y(-1). The user has the option of providing the values of x(ns) in the input file or generating the sequence through use of the subroutine XGEN. All of the aforementioned inputs must be provided for each difference equation to be solved.

The program consists of the main program DIFFEQ.FOR and the subroutines DIFFEQ and XGEN. Flowcharts of the main program and the subroutine DIFFEQ are provided in Appendix I. The program is a computer implementation of the psuedocode algorithms presented in Reference 1 [pp. 84-86]. Also presented in Appendix I are two example problems that demonstrate the capabilities of DIFFEQ.FOR.

10. STATEQ.FOR

The final problem solving program is designed to compute the iterative solution to a set of linear, time invariant state equations. The state and output equations are assumed to be of the form:

$$v(ns+1) = Av(ns) + Bx(ns) \tag{3.10}$$

$$y(ns) = Cv(ns) + Dx(ns) \tag{3.11}$$

where:

* $\mathbf{x}()$ is the M x 1 input vector,
* $\mathbf{v}()$ is the N x 1 state vector,
* $\mathbf{y}()$ is the Q x 1 output vector,
* **A** is an N x N matrix of real constants,
* **B** is an N x M matrix of real constants,
* **C** is an Q x N matrix of real constants and,
* **D** is an Q x M matrix of real constants.

The program will compute the solution to the system of equations for values of ns in the range $0 \le ns \le nstop$. The limits on the parameters M, N, Q, and nstop are:

$$\begin{aligned} &0 \le M \le 4\\ &0 \le N \le 10\\ &0 \le Q \le 4\\ &0 \le nstop \le 99. \end{aligned} \tag{3.12}$$

These parameters as well as the values for the matrices **A**, **B**, **C**, **D**, and the vector comprising the initial condition of the system (vector **v**() at ns = 0) must be provided by the user in the input file. The user can elect to provide values for the input vector **x**(ns) in the input file, or, alternatively, may choose to generate these values by writing the appropriate Fortran statements into subroutine XGEN. The output of the program is the time-history of the vector **y**(ns); however, the program stores all values of the vectors **x**(ns), **v**(ns), and **y**(ns) in the tabular output file STATEQ.OUT.

The program consists of the main program STATEQ.FOR and the subroutines ITRATE and XGEN. The algorithm is a Fortran implementation of a design adapted from Reference 1 [pp.762-765]. The main program reads the input parameters from the input file and calls subroutine ITRATE to compute the solution to the state equations. Subroutine XGEN exists for the sole purpose of allowing the user the option of generating the input sequence(s) internally, rather than providing the values in the input file. Appendix J includes the software flowcharts of the main program and the subroutine ITRATE. Also included in this appendix are two example problems that demonstrate the capabilities of STATEQ.FOR.

B. PLOTTING PROGRAM

1. PLOTDAT.FOR

The sole purpose of the program PLOTDAT.FOR is to create 2-dimensional (2-D) graphs of values read from an input file. The program prompts the user for the name of the input file and will create up to nine 2-D plots, each consisting of up to 999 data points. Each plot requires three labels:

1) the title of the plot,
2) the x-axis label and,
3) the y-axis label.

For plots that consist of more than 25 data points, the program displays the output by connecting the points through use of a linear interpolation (straight-line) scheme. Plots of 25 points or less consist of the symbol '+' at the tabulated points only. The number of data points comprising a given plot, and the plot labels comprise the header information required for each plot. In addition to these parameters, the ordinate and abscissa values for each point to be plotted must be included in the input file. PLOT-DAT.FOR reads these values according to the format: f12.0, 2x, f12.0. The first entry corresponds to the ordinate value and the second entry, the abscissa value. The F-format descriptor was chosen because its use permits values written using either the E or F-format descriptors to be read from the file.

The program consists of the main program PLOTDAT.FOR and the subroutines SCALE and GRIDD. The main program reads the input file and creates the plots. Subroutine SCALE is called by the main program to scale the input values so as to optimize the clarity of the plots. Subroutine GRIDD will overlay a dashed-line grid onto the plot if the user elects to have this done. PLOTDAT.FOR requires a Color Graphics Adapter (CGA) card to display the plots on the monitor screen. In addition, the user can elect to have a printed hardcopy of each plot created. However, to facilitate printing of the graphs, the system in use must include a dot matrix printer. The program will not drive plotters of all types. If the system has an Extended Graphics Adapter (EGA) card, rather than the specified CGA card, hardcopy printouts of the graphs cannot be created by the program directly.

The plots included in Appendices A-J were created using PLOTDAT.FOR. Appendix K is a software flowchart describing the structure of PLOTDAT.FOR and the subroutines SCALE and GRIDD.

IV. Conclusions and Recommendations

As the final phase of this project, the eleven programs included in the package were distributed on a voluntary basis to students enrolled in digital signal processing courses. As part of the course requirements, the students had to solve a variety of signal processing problems representative of the type that the programs were designed for. Throughout this software evaluation phase, the students provided feedback as to the utility of the programs. While a majority of this feedback was positive, three areas of concern warrant attention in this report.

1. The graphics capability of PLOTDAT.FOR is limited to machines with CGA/EGA graphics cards.

2. In order to get the most use out of the ten problem solving programs, the user must have access to a Fortran compiler capable of compiling the programs as written.

3. Because the programs are file-driven, the user must carefully read the header text of each program in order to execute the programs successfully. Although the example problems in the header text and the corresponding sample input files seem to alleviate some of the common formatting errors, new users experienced some displeasure with the format requirements.

The first area of concern, although valid, is consistent with the advertised capabilities of PLOTDAT.FOR. As graphics software becomes more advanced, PLOTDAT.FOR should be updated to incorporate any changes that increase the portability of the program. A caveat to this, however, is the obvious temptation to use sophisticated software designed for more capable machines at the expense of its compatibility with less capable ones. The primary goal of the plotting program is portability among the broadest possible span of target users.

The second area of concern is also somewhat warranted. The programs were optimally designed for use on a machine equipped with a suitable Fortran compiler. The presence of a compiler allows the user to add source code to the programs, an option particularly useful in generating long sequences of input data. To this end, however, any high level language can be used to create the input files, as long as the required data can be stored in a form compatible with the programs. In brief, any Fortran compiler would be suitable for generating the input data thus eliminating the need for a specific compiler.

Lastly, the dissatisfaction among the students over the file-driven versus menu-driven design of the programs may warrant a future design change. The principle concern was the rather stringent input formats required by the programs. As the students became more experienced with the programs, however, these problems subsided somewhat. Nevertheless, the programs' input sections can be restructured to incorporate the features of both the menu and the file-driven designs. As envisioned, in menu-driven mode the programs would prompt the user for each input value, storing the values in an input file for future use. This mode, despite its time consuming mechanics, would be attractive to first-time users who could use the input files created by the programs as a guide for subsequent runs. Experienced users would create the required input files and run the programs in the file-driven mode. This mode is already incorporated in the programs as they exist. The redesign of the programs; therefore, would only require incorporation of a menu-driven mode. Such a design change is fully within the capabilities of the Fortran compiler used to create these programs.

APPENDIX A

Two digital filter designs are developed in this section to demonstrate the performance of the program DIGFREQ.FOR. An analysis of the designs includes a listing of the input required to execute the program and the corresponding output that is produced. The plotting program PLOTDAT.FOR was used to plot the output data and hard copies of these plots are also included. The software flowchart of the program is included as the last pages of this appendix.

The variable names listed below are used in the Fortran source code of DIGFREQ.FOR and in the corresponding flowcharts.

numsys -	The integer value that specifies the number of distinct filter equations whose parameters occur in the input file.
L -	The integer value that specifies the order of the numerator polynomial.
N -	The integer value that specifies the order of the denominator polynomial.
dsorce -	The character string 'F' or 'S' denoting whether the system coefficients are to be read from the input file (F) or generated (S) through use of the subroutine COEFF.
theta0 -	The starting value of Θ (rad).
numpts -	The integer value that specifies the desired number of frequency points.
yscal -	The character string 'STD' or 'LOG' that specifies whether standard magnitude (STD) or magnitude expressed in decibels (LOG) is to be computed.
b() -	The array containing the numerator coefficients.
c() -	The array containing the denominator coefficients.
mh() -	The array containing the magnitude values of the computed frequency response.
ph() -	The array containing the phase values (degrees) of the computed frequency response.

Example #1

This example is identical to the sample problem found in the header text of the program. The system is a first order low-pass filter with a pole at z = 0.5, and a zero at z = 0.0. The filter transfer function, in the form of Equation (3.1), is:

$$H(z) = \frac{z}{z - .5} \tag{A.1}$$

The goal is to calculate the frequency response of the filter for frequencies in the range: $0 \leq \Theta \leq 3.14159$ (rad). The listings that follow include the input file DIGFREQ.TST required to produce 11 output points and the tabular output file DIGFREQ.OUT. Also included are plots of the output for 101 frequency points. An analysis of the data confirms the low-pass nature of the filter.

DIGFREQ.TST

```
1
001        001        F        STD
.314159    0.0        011
1.0        0.0
1.0        -.5
```

DIGFREQ.OUT

```
                    INPUT DATA FOR SYSTEM # 1

INPUT DATA SOURCEFILE: DIGFREQ.TST
DEGREE OF NUMERATOR =   1
DEGREE OF DENOMINATOR =   1
dsorce = F
NUMBER OF FREQUENCY POINTS =  11     MAGNITUDE OPTION = STD
STARTING VALUE OF THETA =  .000000E+00
INCREMENT OF THETA =  .314159E+00

THE NUMERATOR COEFFICIENTS b(0),b(1)...b(L) ARE:

 .1000E+01    .0000E+00


THE DENOMINATOR COEFFICIENTS c(0),c(1)...c(N) ARE:

 .1000E+01   -.5000E+00

                    OUTPUT DATA FOR SYSTEM # 1

     THETA            MAGNITUDE          PHASE
    (RADIANS)                          (DEGREES)
   .000000E+00      .200000E+01       .000000E+00
   .314159E+00      .182897E+01      -.164149E+02
   .628318E+00      .150588E+01      -.262677E+02
   .942477E+00      .122886E+01      -.298071E+02
   .125664E+01      .103088E+01      -.293546E+02
   .157080E+01      .894428E+00      -.265651E+02
   .188495E+01      .800894E+00      -.223862E+02
   .219911E+01      .737654E+00      -.173608E+02
   .251327E+01      .696900E+00      -.118186E+02
   .282743E+01      .674038E+00      -.597793E+01
   .314159E+01      .666667E+00      -.484184E-04

  -------------  END OF RUN, SYSTEM # 1  -------------
```

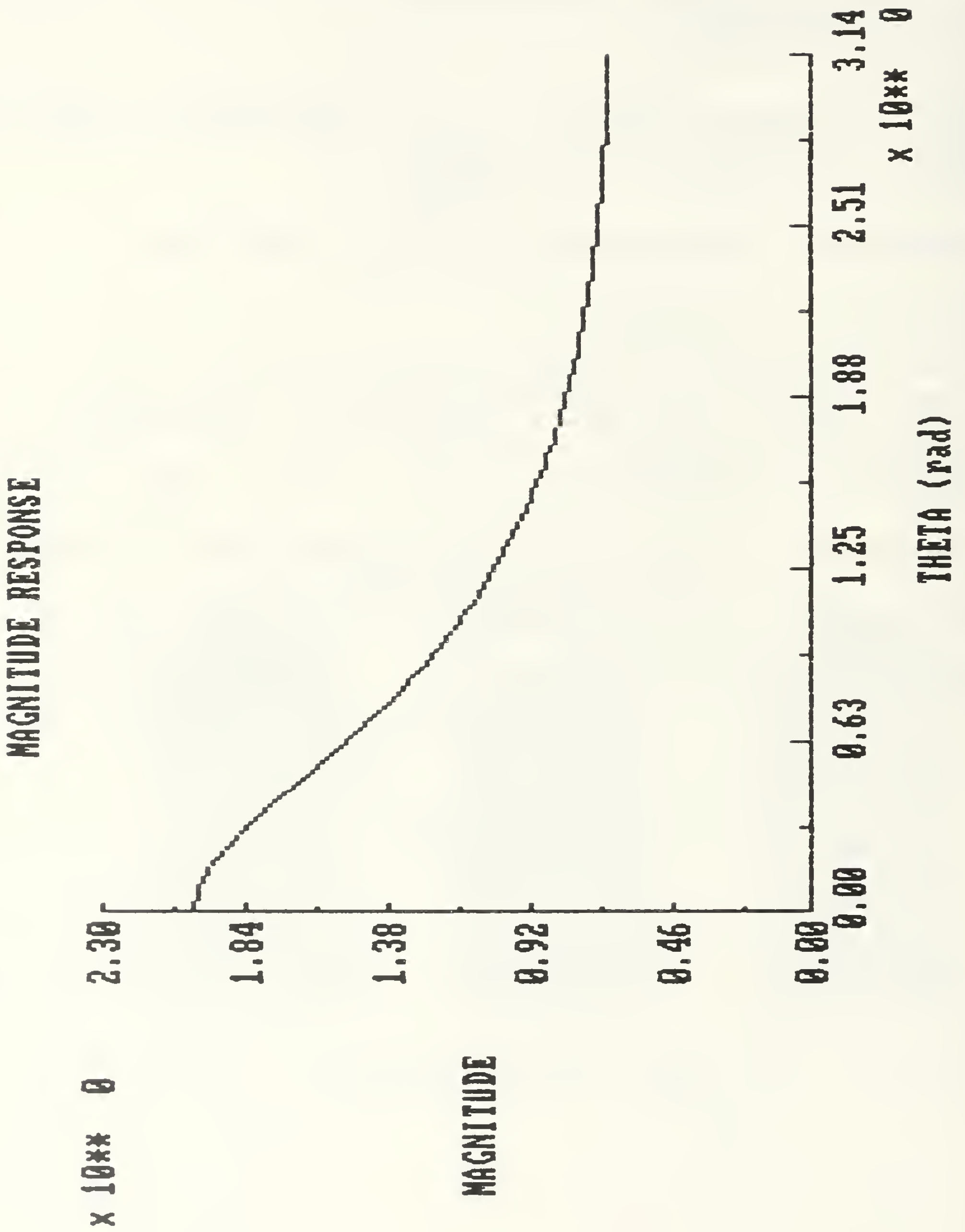


Figure A.1 Magnitude response of a low-pass filter - Example #1.

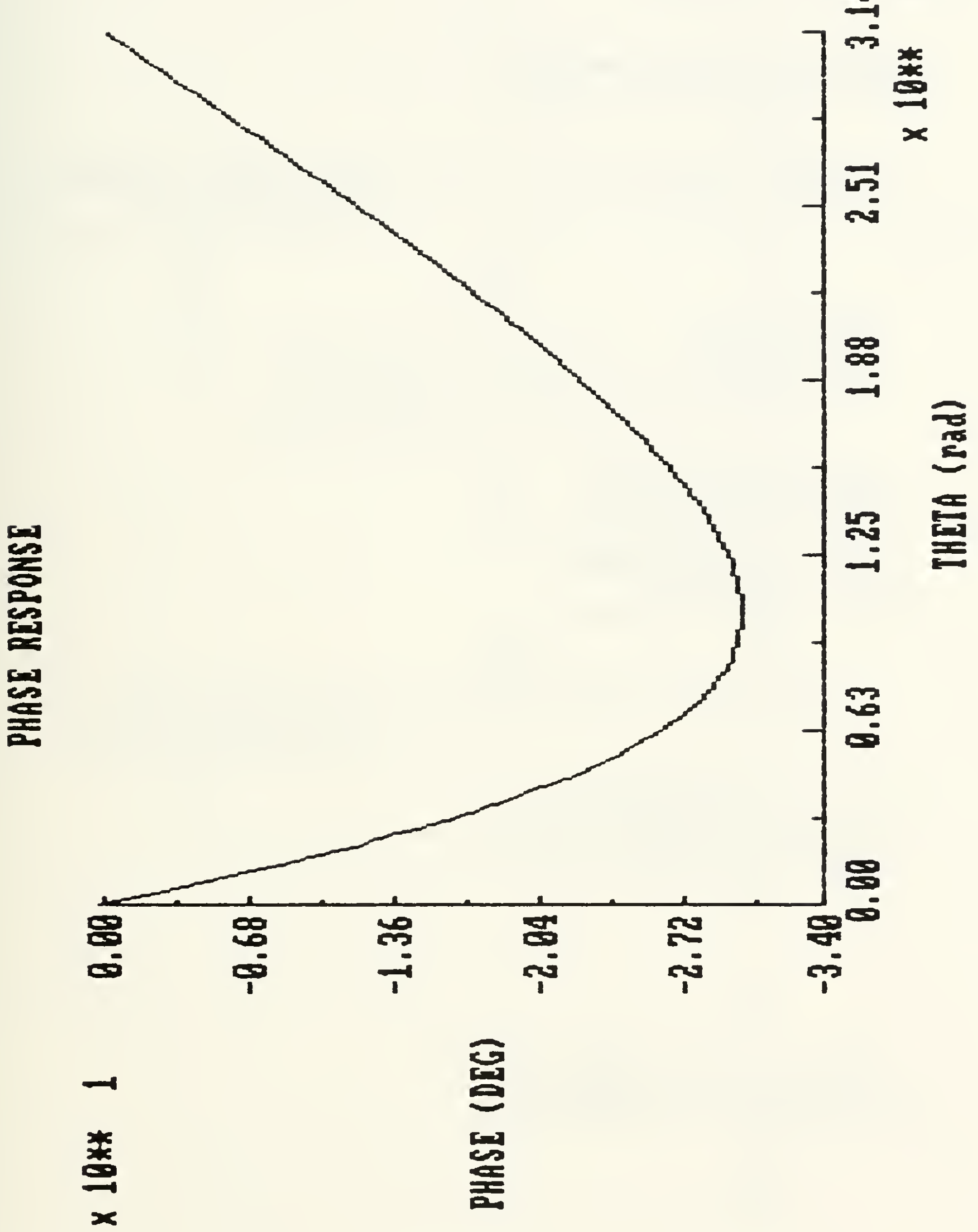


Figure A.2 Phase response of a low-pass filter - Example #1.

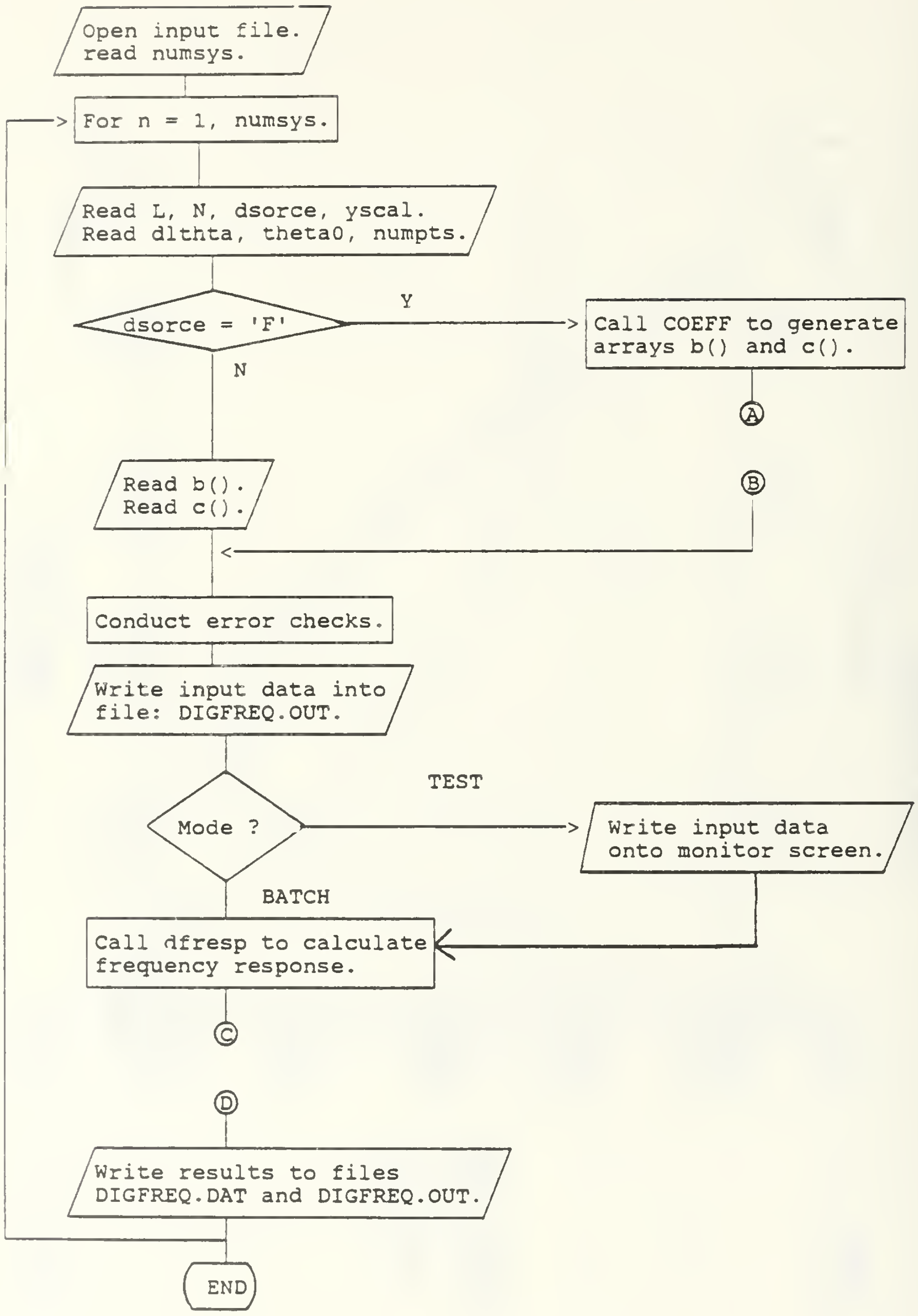

Open input file.
read numsys.
For n = 1, numsys.
Read L, N, dsorce, yscal.
Read dlthta, theta0, numpts.
dsorce = 'F'
Y
N
Call COEFF to generate arrays b() and c().
A
B
Read b().
Read c().
Conduct error checks.
Write input data into file: DIGFREQ.OUT.
Mode ?
TEST
BATCH
Write input data onto monitor screen.
Call dfresp to calculate frequency response.
C
D
Write results to files DIGFREQ.DAT and DIGFREQ.OUT.
END

Figure A.3 DIGFREQ.FOR Software Flowchart.

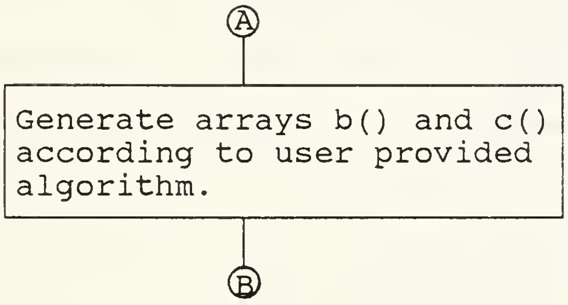

A
Generate arrays b() and c() according to user provided algorithm.
B

Figure A.4 COEFF Subroutine Flowchart.

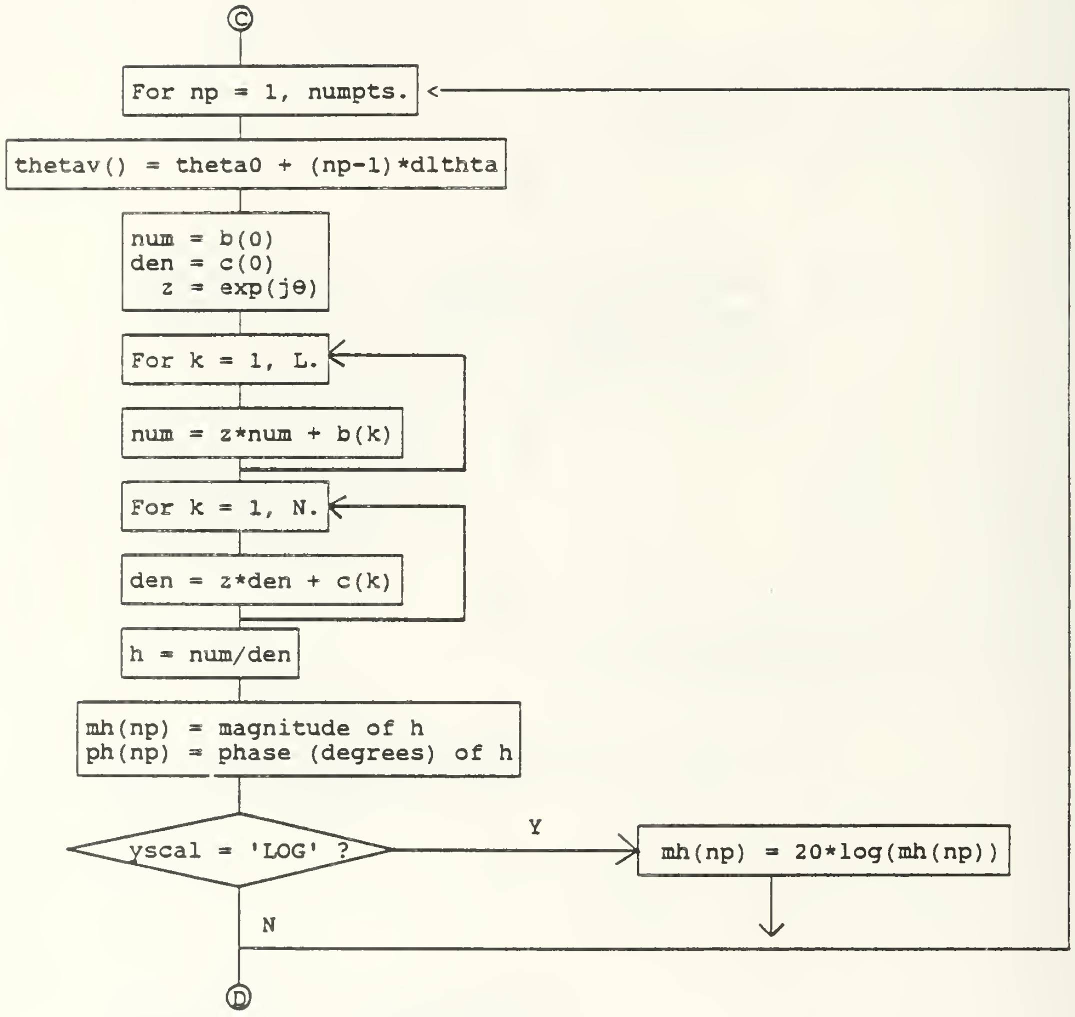


Figure A.5 DFRESP Subroutine Flowchart.

Appendix B

A fifth-order, low-pass filter is used in this appendix to demonstrate the performance of the program ANLGFREQ.FOR. The filter transfer function is in the form of Equation (3.2). The analysis that follows compares the theoretical frequency design specifications of the filter to the frequency response computations produced by the program. The software flowcharts of the program ANLGFREQ.FOR and the subroutine AFRESP are included as the last pages of this appendix.

The variable names listed below are used in the Fortran source code of the program and in the corresponding flowcharts.

numsys -	The integer value that specifies the number of distinct filter equations whose parameters occur in the input file.
L -	The integer value that specifies the order of the numerator polynomial.
N -	The integer value that specifies the order of the denominator polynomial.
omega0 -	The starting value of w (rad/s) for which the frequency response is to be calculated.
dlomga -	The increment of w (rad/s).
numpts -	The integer value that specifies the desired number of frequency points.
yscal -	The character string 'STD' or 'LOG' that specifies whether standard magnitude (STD) or magnitude expressed in decibels (LOG) is to be computed.
b() -	The array containing the numerator coefficients.
a() -	The array containing the denominator coefficients.
mh() -	The array containing the computed magnitude value for each frequency point.
ph() -	The array containing the computed phase value (degrees) for each frequency point.

Example #1

A fifth-order Chebyshev Low-Pass Filter has the transfer function:

$$H(s) = \frac{62268.8}{s^5 + 10.605s^4 + 337.5s^3 + 2342.25s^2 + 23236.9s + 62268.8}$$

(B.2)

The filter was designed to have a ripple passband edge frequency of w = 15.0 (rad/s) and a maximum ripple of 2 dB [Ref. 1:pp. 630-637]. The input file ANLGFREQ.IN listed below provides the inputs necessary for ANLGFREQ.FOR to compute the magnitude (dB) and phase (Deg) response of this filter across the frequency range: $0 \le w \le 20$ (rad/s).

As can be seen from both the tabular output and the accompanying plots the filter specifications have been met. The magnitude of 0 db at frequency w = 0 is characteristic of this type of normalized low-pass filter. The edge of the ripple passband has a magnitude of -2 dB at w = 15 (rad/s) and the 2 dB ripple is not exceeded within the ripple passband.

ANLGFREQ.IN

```
1
000        005        021        LOG
1.0        0.0
62268.8
1.0        10.605     .3375E03   2342.25    23236.9    62268.8
```

ANLGFREQ.OUT

```
                    INPUT DATA FOR SYSTEM # 1

INPUT DATA SOURCEFILE: ANLGFREQ.IN
DEGREE OF NUMERATOR =    0
DEGREE OF DENOMINATOR =    5
NUMBER OF FREQUENCY POINTS =  21      MAGNITUDE OPTION = LOG
STARTING VALUE OF OMEGA =  .000000E+00
INCREMENT OF OMEGA =  .100000E+01

THE NUMERATOR COEFFICIENTS b(0),b(1)...b(L) ARE:

 .6227E+05


THE DENOMINATOR COEFFICIENTS a(0),a(1)...a(N) ARE:

 .1000E+01      .1060E+02      .3375E+03      .2342E+04
 .2324E+05    .6227E+05


                    OUTPUT DATA FOR SYSTEM # 1

        OMEGA         MAGNITUDE(dB)        PHASE
       (rad/s)                           (DEGREES)
    .000000E+00      .000000E+00      .000000E+00
    .100000E+01    -.260284E+00     -.209105E+02
    .200000E+01    -.868081E+00     -.395377E+02
    .300000E+01    -.149418E+01     -.553515E+02
    .400000E+01    -.189199E+01     -.691887E+02
    .500000E+01    -.193554E+01     -.823632E+02
    .600000E+01    -.158803E+01     -.963804E+02
    .700000E+01    -.917171E+00     -.112999E+03
    .800000E+01    -.196446E+00     -.133940E+03
    .900000E+01     .423856E-01     -.159062E+03
    .100000E+02    -.519360E+00      .175548E+03
    .110000E+02    -.143810E+01      .153702E+03
    .120000E+02    -.198221E+01      .134785E+03
    .130000E+02    -.158172E+01      .114249E+03
    .140000E+02    -.216956E+00      .804525E+02
    .150000E+02    -.201998E+01      .233437E+02
    .160000E+02    -.831394E+01     -.135395E+02
    .170000E+02    -.140297E+02     -.300361E+02
    .180000E+02    -.187218E+02     -.391769E+02
    .190000E+02    -.226985E+02     -.451868E+02
    .200000E+02    -.261739E+02     -.495617E+02

---------------- END OF RUN, SYSTEM #1 ------------------
```

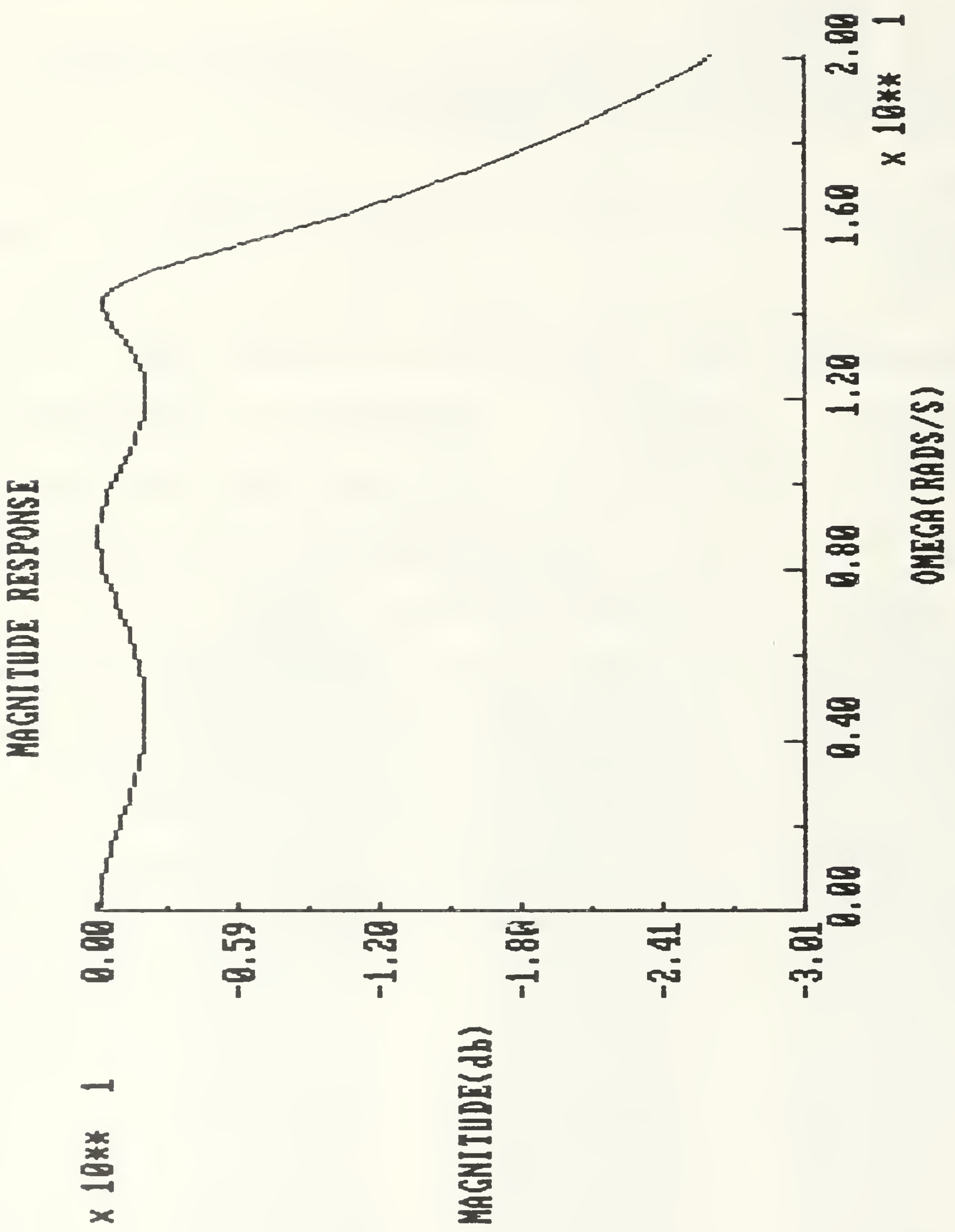


Figure B.1 Magnitude response of a Chebyshev Low-Pass Filter - Example #1.

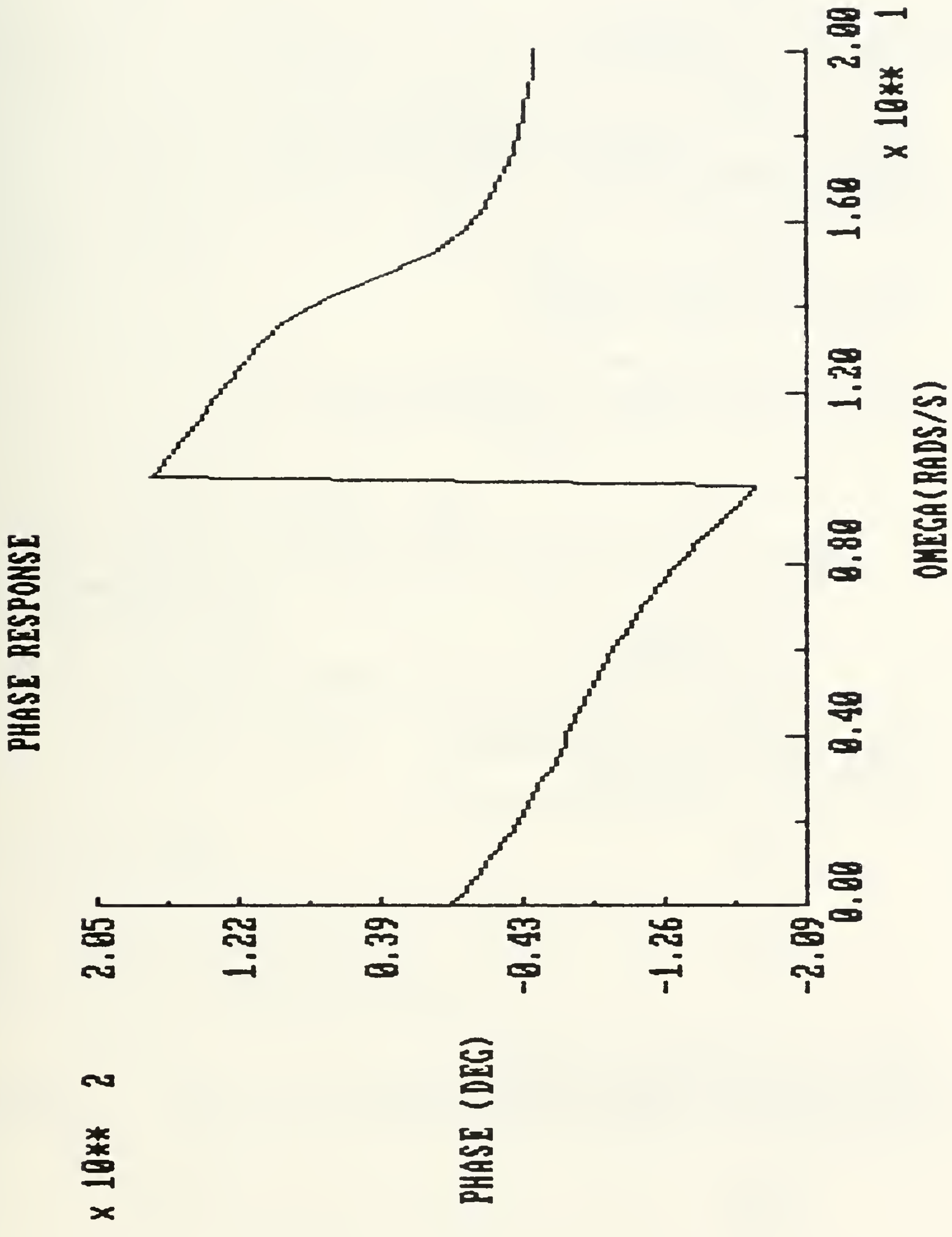
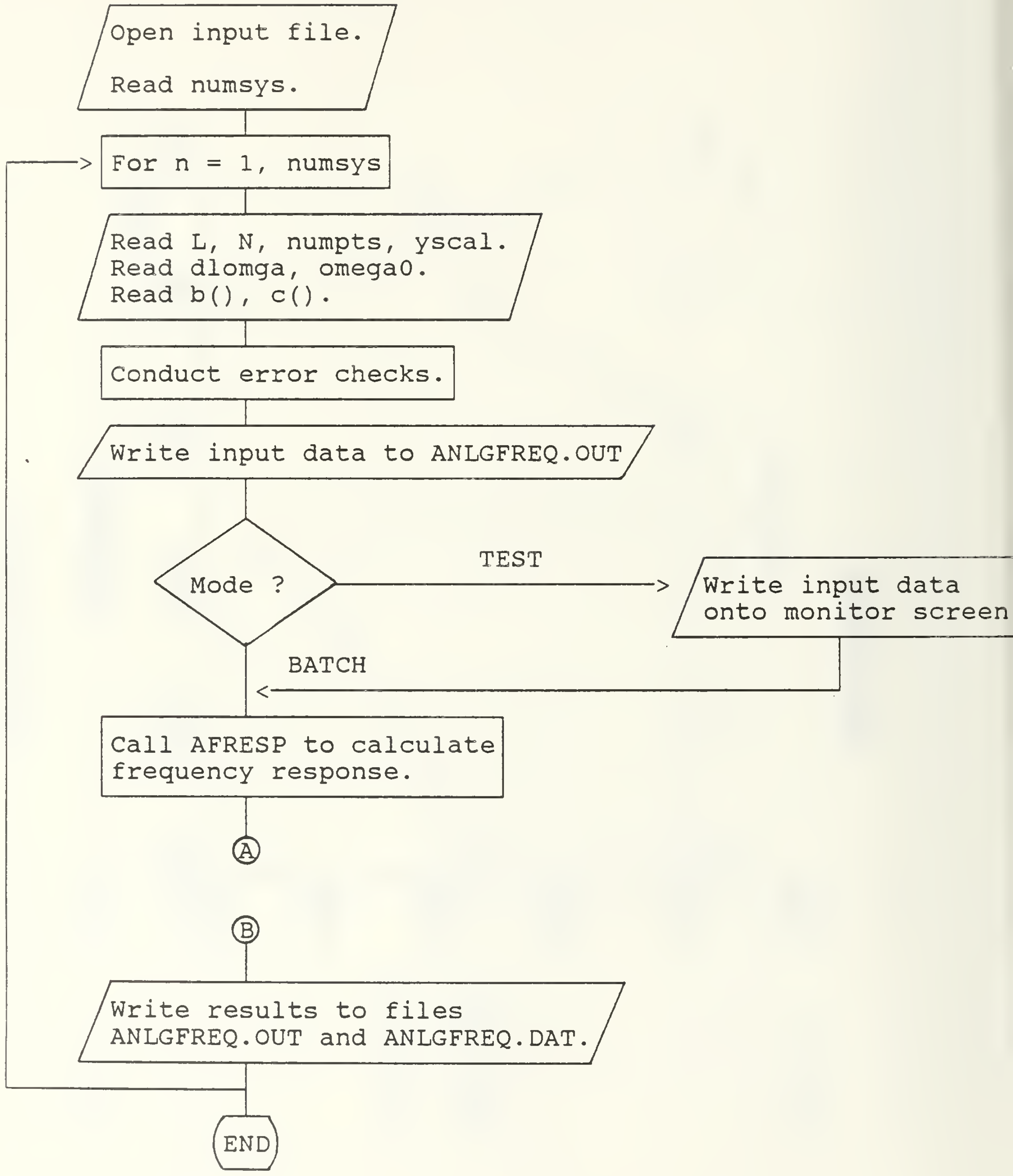


Figure B.2 Phase response of a Chebyshev Low-Pass Filter -Example #1.



Open input file.
Read numsys.
For n = 1, numsys
Read L, N, numpts, yscal.
Read dlomga, omega0.
Read b(), c().
Conduct error checks.
Write input data to ANLGFREQ.OUT
Mode ?
TEST
Write input data onto monitor screen
BATCH
Call AFRESP to calculate frequency response.
A
B
Write results to files ANLGFREQ.OUT and ANLGFREQ.DAT.
END

Figure B.3 ANLGFREQ.FOR Software Flowchart.

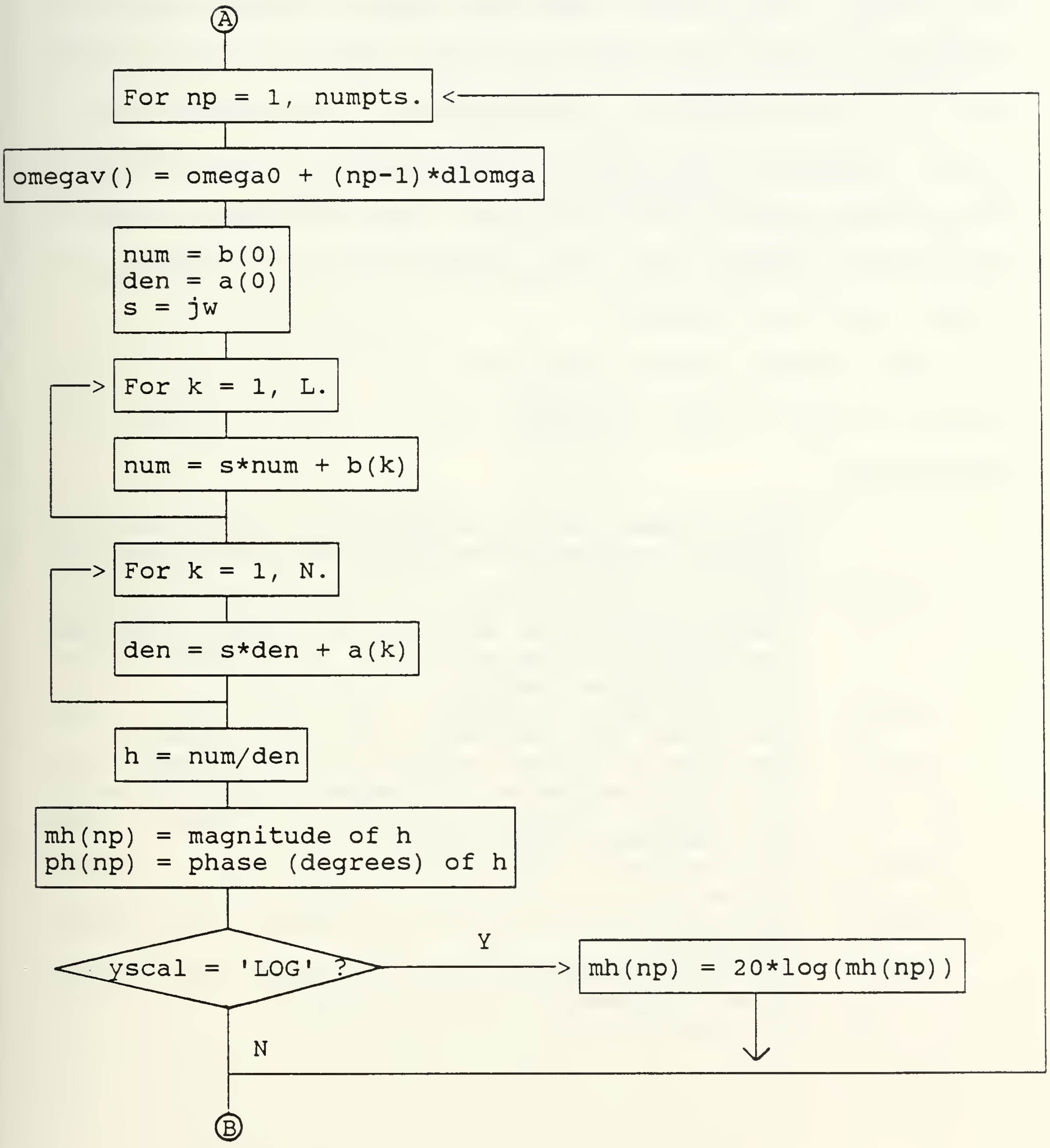


Figure B.4 AFRESP Subroutine Flowchart.

Appendix C

As a practical exercise to demonstrate the performance of DFT.FOR, two sample sequences were generated and the program DFT.FOR was used to compute either the DFT or the IDFT of the sequences. Contained in this appendix is a brief analysis of the chosen examples including listings of the output produced by the program. The software flowcharts describing DFT.FOR and the subroutines DFT, INVDFT, and SAMPLE are also included.

The variable names listed below are used in the Fortran source code of the program and in the corresponding flowcharts.

N -	The integer value that specifies the number of complex samples contained in the input sequence xin().
dsorce -	The character string 'F' or 'S' that specifies whether the input data is to be read from the input file (F) or generated (S) through use of the subroutine SAMPLE.
option -	The character string 'DFT' or 'INV' that specifies the computation to be performed.
xin() -	The complex array containing the N samples of the input sequence. A sequence consisting of only 'real' numbers is stored as values having an 'imaginary' part of 0.0.
xout() -	The complex array containing the N output values.
xmag() -	The array containing the N values of the output magnitude.
xph() -	The array containing the N values of the output phase (Degrees).
wm -	The complex value: $wm = e^{-j2\pi k/N}$.

Example #1

For the first example problem a unit ramp sequence consisting of 5 values was input to DFT.FOR. The goal was to compute the DFT of the sequence. This example problem is also developed in the header text of DFT.FOR and can be run by the user by selecting Test Mode and entering 'DFT.TST' when prompted for the name of the input file. The listings that follow include the input file DFT.TST and the tabular output file DFT.OUT.

DFT.TST

```
005        F          DFT
0.0        0.0
1.0        0.0
2.0        0.0
3.0        0.0
4.0        0.0
```

DFT.OUT

```
INPUT DATA SOURCEFILE: DFT.TST
VALUE OF N =    5     dsorce = F     option = DFT


             INPUT DATA

  SAMPLE #       REAL              IMAGINARY
     0         .000000E+00       .000000E+00
     1         .100000E+01       .000000E+00
     2         .200000E+01       .000000E+00
     3         .300000E+01       .000000E+00
     4         .400000E+01       .000000E+00



             OUTPUT DATA

SAMPLE #      REAL          IMAGINARY        MAGNITUDE          PHASE
                                                              (DEGREES)
   0       .100000E+02     .000000E+00      .100000E+02      .000000E+00
   1      -.250000E+01     .344096E+01      .425325E+01      .126000E+03
   2      -.250000E+01     .812300E+00      .262866E+01      .162000E+03
   3      -.250000E+01    -.812299E+00      .262866E+01     -.162000E+03
   4      -.250000E+01    -.344096E+01      .425326E+01     -.126000E+03
```

Example #2

As an extension to the first example problem and to demonstrate the IDFT option, the DFT results of the first problem were input to the program and the IDFT was computed. As one would expect, the original unit ramp sequence was generated confirming the ability of the program to compute either the DFT or its inverse, the IDFT. Numerical roundoff corresponding to the single precision accuracy of DFT.FOR accounts for the slight deviation between the original unit ramp sequence and the results produced by this example.

DFT.IN

```
005       F         INV
.10000E02 0.0
-.2500E01 .344096E01
-.2500E01 .812300
-.2500E01 -.812299
-.2500E01 -3.44096
```

DFT.OUT

```
INPUT DATA SOURCEFILE: DFT.IN
VALUE OF N =  5     dsorce = F    option = INV


            INPUT DATA

  SAMPLE #      REAL              IMAGINARY
     0         .100000E+02      .000000E+00
     1        -.250000E+01      .344096E+01
     2        -.250000E+01      .812300E+00
     3        -.250000E+01     -.812299E+00
     4        -.250000E+01     -.344096E+01




            OUTPUT DATA

SAMPLE #     REAL           IMAGINARY        MAGNITUDE          PHASE
                                                             (DEGREES)
   0       .000000E+00     .238419E-06     .238419E-06     .900000E+02
   1       .999998E+00    -.210175E-06     .999998E+00    -.120422E-04
   2       .200000E+01     .136523E-06     .200000E+01     .391109E-05
   3       .300000E+01     .358854E-06     .300000E+01     .685361E-05
   4       .400000E+01     .298563E-06     .400000E+01     .427659E-05
```

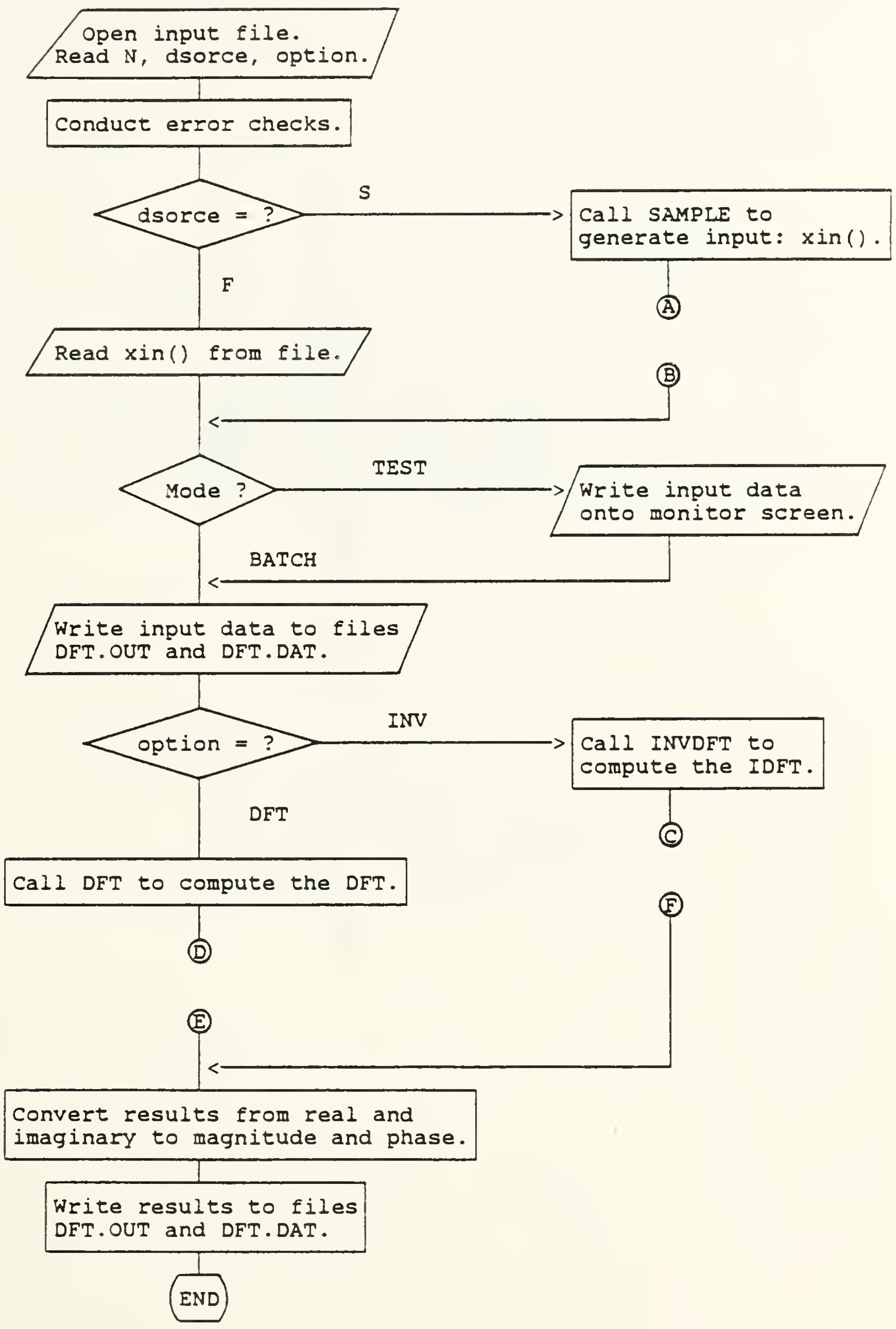

Open input file.
Read N, dsorce, option.
Conduct error checks.
dsorce = ?
S
Call SAMPLE to generate input: xin().
A
F
Read xin() from file.
B
Mode ?
TEST
Write input data onto monitor screen.
BATCH
Write input data to files DFT.OUT and DFT.DAT.
option = ?
INV
Call INVDFT to compute the IDFT.
C
DFT
Call DFT to compute the DFT.
F
D
E
Convert results from real and imaginary to magnitude and phase.
Write results to files DFT.OUT and DFT.DAT.
END

Figure C.1 DFT.FOR Software Flowchart.

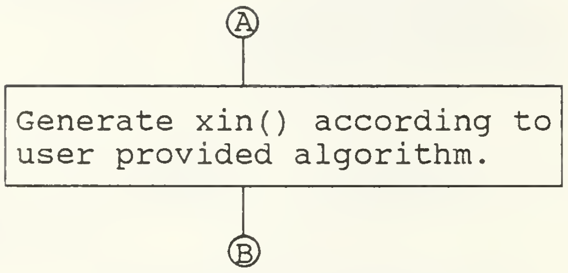


Figure C.2 SAMPLE Subroutine Flowchart.

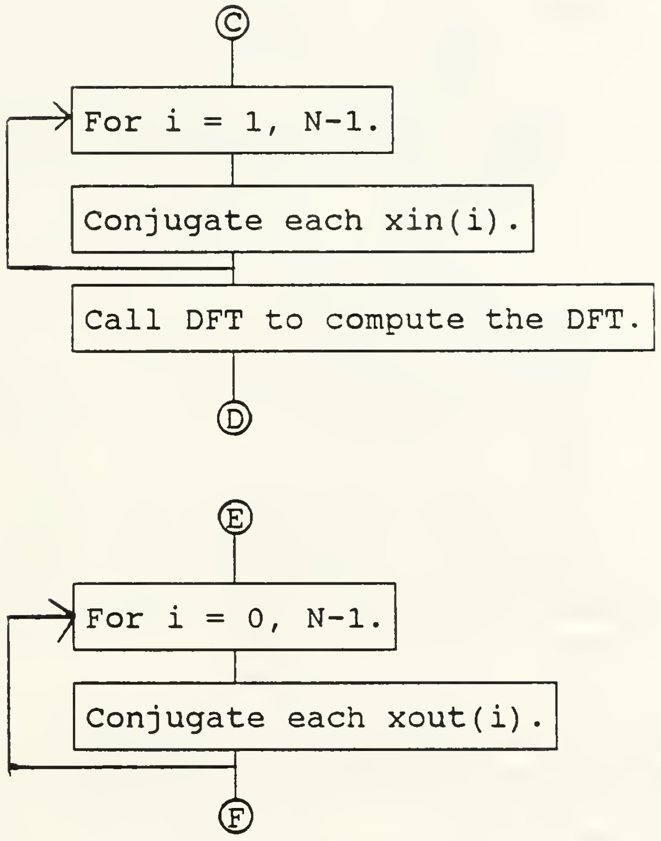


Figure C.3 INVDFT Subroutine Flowchart.

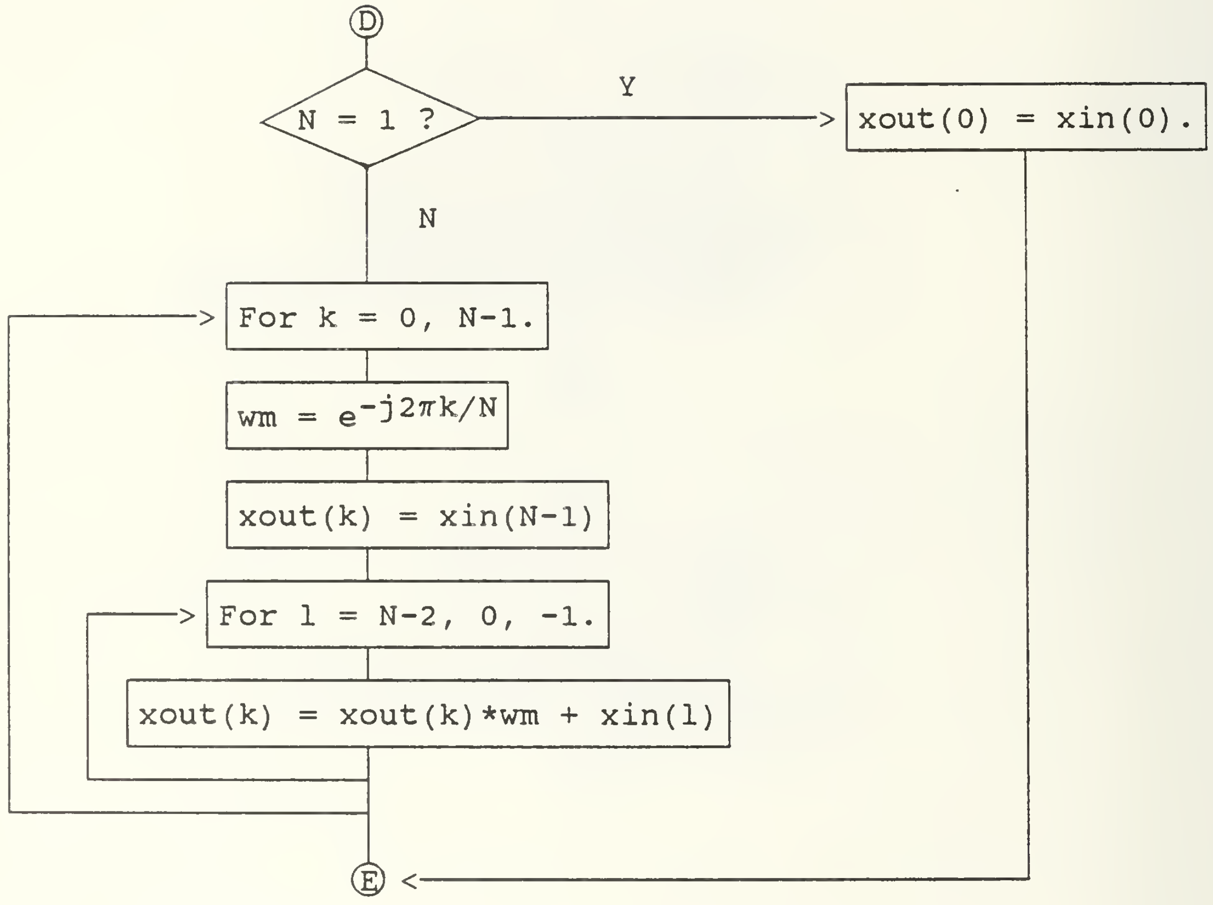


Figure C.4 DFT Subroutine Flow Chart.

Appendix D

Three example problems are developed in this appendix to demonstrate the program PRDGRM.FOR. Example #1 is a demonstration of the program using a short input sequence. A listing of both the input sequence and the output sequence are included in the analysis. Examples #2 and 3 require long sequences of data and therefore only plots of the input and output sequences are included. The software flowchart of PRDGRM.FOR is included as the last page of this appendix. Since the flowcharts of the subroutines SAMPLE and DFT were presented in Appendix C, they are not repeated in this appendix.

The variable names listed below are used in the Fortran source code of PRDGRM.FOR and in the corresponding flow-charts.

N -	The integer value that specifies the number of complex samples contained in the array xn().
dsorce -	The character string 'F' or 'S' that specifies whether the input data is to be read from the input file (F) or generated (S) through use of the subroutine SAMPLE.
yscal -	The character string 'STD' or 'LOG' that specifies whether the output sequence is to be expressed as standard (STD) or decibel (LOG) magnitude.
xn() -	The complex array containing the N input values. A sequence consisting of only real numbers is stored as values having an imaginary part of 0.0.
xk() -	The complex array containing the DFT sequence corresponding to xn(), i.e., xk() = DFT[xn()].
Sxx() -	The array containing the N values of the periodogram sequence. This array contains the output sequence of the program.

Example #1

Because PRDGRM.FOR uses the DFT algorithm as part of the periodogram computation, a simple demonstration of the program's accuracy is to compare the results of PRDGRM.FOR to the results of DFT.FOR using the same input sequence for both programs. The sequence chosen was the five samples of the unit ramp. The listings that follow include the input file PRDGRM.TST required to run this problem, as well as the tabular output file PRDGRM.OUT containing the computed results. The DFT results were presented in Appendix C.

By comparing the output sequences of the two programs it is easy to see the relationship between the DFT and the periodogram as described by Equation (3.6).

This example problem also appears in the header text of PRDGRM.FOR. The user can run this problem by selecting Test Mode and entering 'PRDGRM.TST' as the input file name.

PRDGRM.TST

```
005         F           STD
0.0         0.0
1.0         0.0
2.0         0.0
3.0         0.0
4.0         0.0
```

PRDGRM.OUT

```
INPUT DATA SOURCEFILE: PRDGRM.TST
VALUE OF N =    5     dsorce = F     MAGNITUDE OPTION = STD


                INPUT DATA
                   xn()

      n      REAL          IMAGINARY
      0     .0000E+00     .0000E+00
      1     .1000E+01     .0000E+00
      2     .2000E+01     .0000E+00
      3     .3000E+01     .0000E+00
      4     .4000E+01     .0000E+00


              OUTPUT DATA

     k        Sxx(k)
     0     .2000E+02
     1     .3618E+01
     2     .1382E+01
     3     .1382E+01
     4     .3618E+01
```

Example # 2

A low-pass filter presented earlier had the transfer function:

$$H(z) = \frac{z}{z - .5} \qquad \text{(D.1)}$$

The impulse response of this filter can be computed iteratively by the corresponding difference equation:

$$\begin{aligned} & y(n) = x(n) + .5y(n-1), \qquad n = 0, 1, \ldots, N-1 \\ \text{where: } & x(n) = 1.0 \quad \text{at } n = 0 \\ & \ 0.0 \quad \text{otherwise} \\ & y(-1) = 0.0 \end{aligned} \qquad \text{(D.2)}$$

The performance of PRDGRM.FOR can be evaluated by computing the periodogram of the resulting impulse response and comparing the results to the frequency response of the filter. The frequency response was computed previously using DIGFREQ.FOR and a plot of the frequency response appears in Appendix A.

The subroutine SAMPLE was used to generate 200 samples of the filter's impulse response. Included on the page that follows is a plot of the log periodogram sequence. The results produced by PRDGRM.FOR as well as those produced by DIGFREQ.FOR confirm the low-pass nature of the filter. The disparity of the plots can be attributed somewhat to the implied rectangular windowing of a finite-length sequence used as an input to the periodogram algorithm. Because of this windowing effect, the use of the periodogram as a spectral estimation technique is somewhat limited.

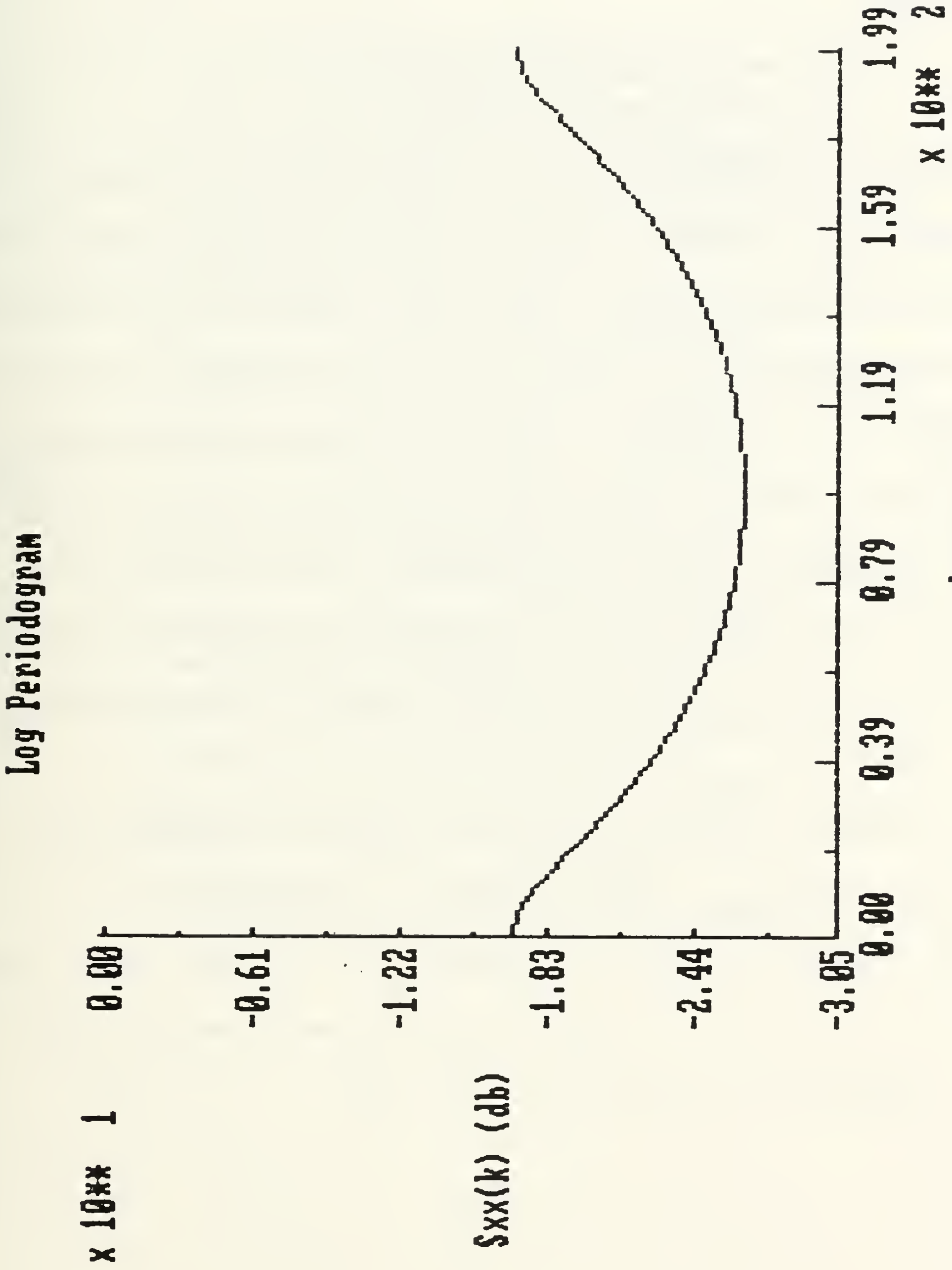


Figure D.1 Periodogram of a low-pass filter's impulse response - Example #2.

Example # 3

As a final demonstration of PRDGRM.FOR, the subroutine SAMPLE was used to generate the input sequence according to the equation:

$$x(n) = 2.0\cos(2\pi n500/5000) \tag{D.3}$$

The sequence consists of N = 200 samples. Plots of the input sequence as well as the output sequence are included on the pages that follow. For this sequence, the frequency of the continuous-time signal is $f = 500$ Hz and the sampling frequency is $f_s = 5000$ Hz. Since 5000 Hz corresponds to $\Theta = 2\pi$ rad, the periodogram should peak at $\Theta = \pi/5$ rad, the digital frequency that corresponds to $f = 500$ Hz. To demonstrate the ability of the program to convert the output to decibels, the input parameter yscal was assigned the value: 'LOG'.

An analysis of the plotted output confirms the anticipated results. The signal, consisting of a pure sinusoid, has a digital frequency of $\pi/5$ rad when sampled at 5000 Hz. The plot of the periodogram remains below 0 dB for all frequencies except $\Theta = \pi/5$ rad.

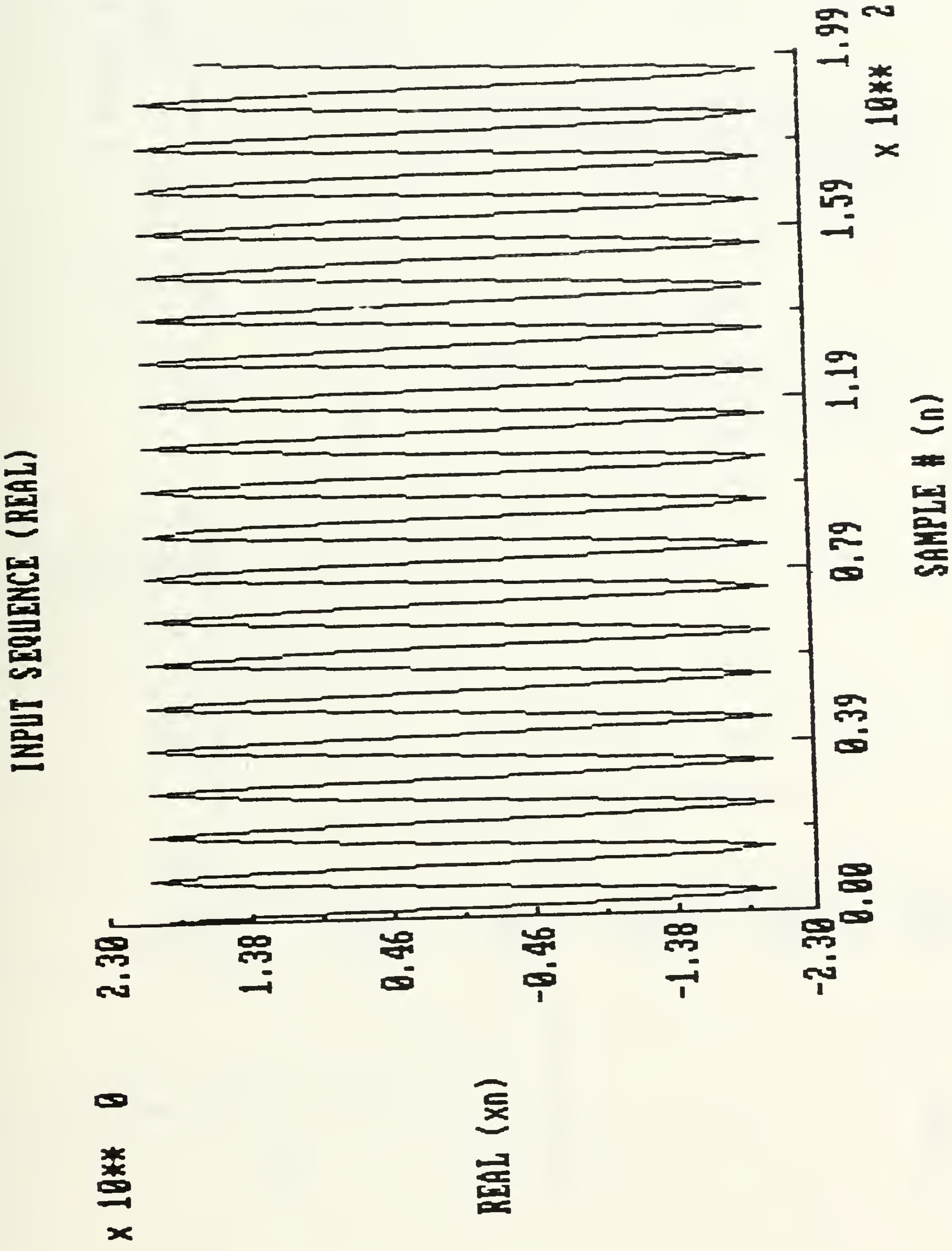


Figure D.2 Input sequence x(n) = 2cos(2πn500/5000) - Example #3.

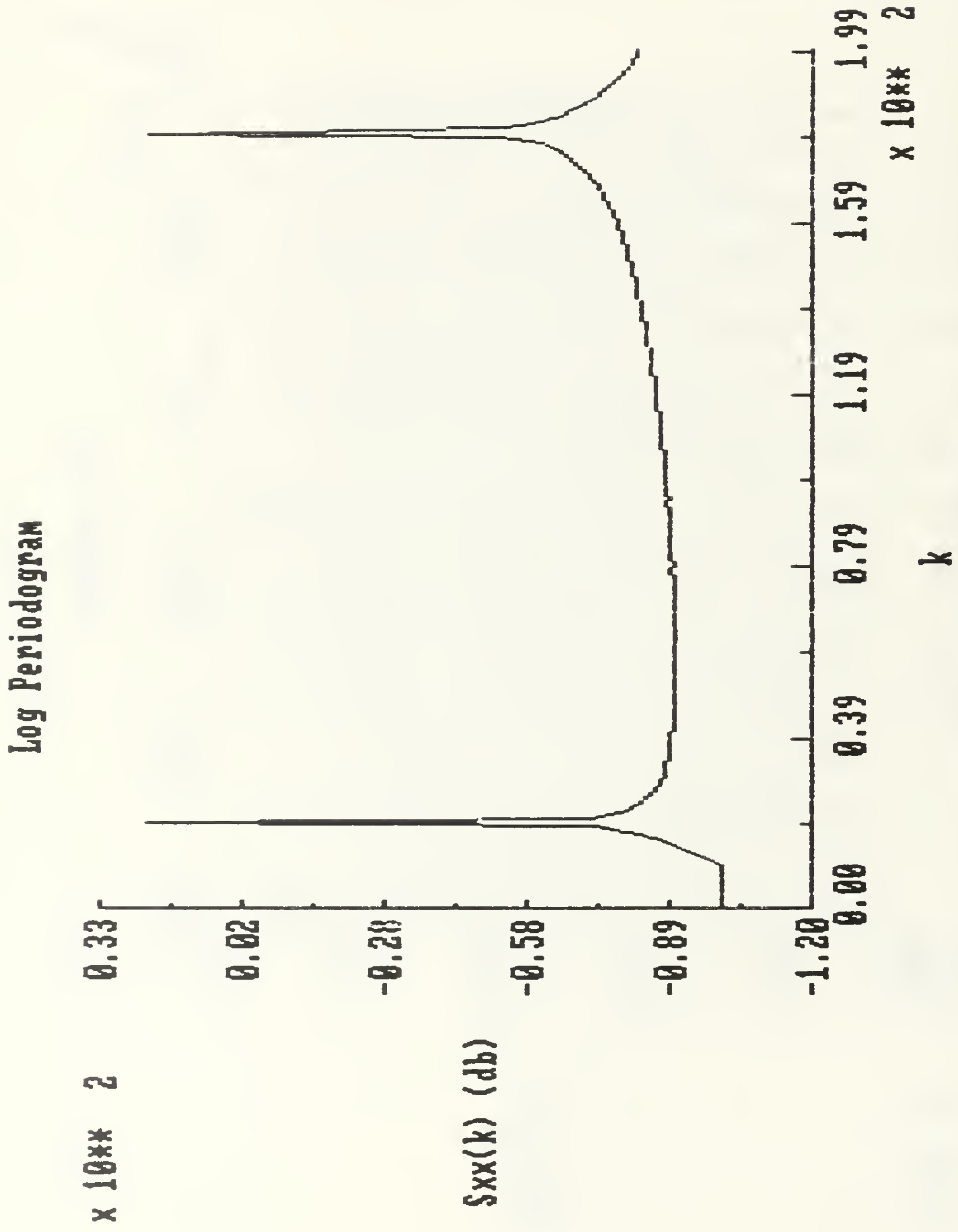


Figure D.3 Periodogram of a sinusoid - Example #3.

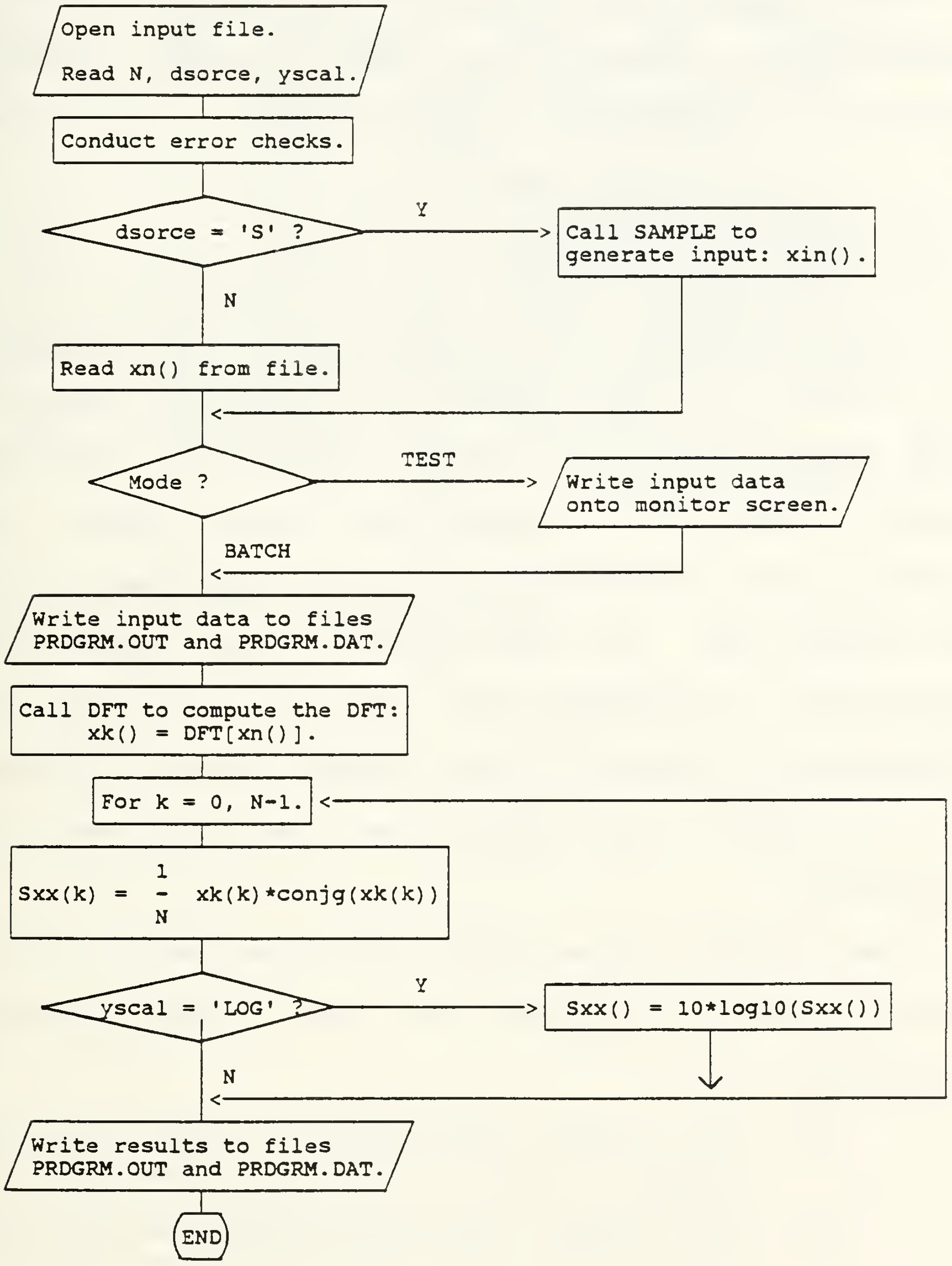

Open input file.
Read N, dsorce, yscal.
Conduct error checks.
dsorce = 'S' ?
Y
Call SAMPLE to generate input: xin().
N
Read xn() from file.
Mode ?
TEST
Write input data onto monitor screen.
BATCH
Write input data to files PRDGRM.OUT and PRDGRM.DAT.
Call DFT to compute the DFT: xk() = DFT[xn()].
For k = 0, N-1.
Sxx(k) = 1/N xk(k)*conjg(xk(k))
yscal = 'LOG' ?
Y
Sxx() = 10*log10(Sxx())
N
Write results to files PRDGRM.OUT and PRDGRM.DAT.
END

Figure D.4 PRDGRM.FOR Software Flowchart.

Appendix E

As a demonstration of the capabilities of CONCORDT.FOR this appendix presents four example problems. Each example problem demonstrates one of the following computations:

1. Circular convolution.
2. Linear convolution.
3. Circular correlation.
4. Linear correlation.

A brief analysis of the output generated by the program is included for each problem. The examples that use short data sequences include tabular listings of the input and output values. The examples that require longer sequences include plots of the input and output sequences as generated by PLOTDAT.FOR. The last pages of this appendix are the flowcharts of CONCORDT.FOR and the subroutine ZEROPAD. Flowcharts of subroutines DFT, INVDFT, SAMPL1 and SAMPL2 are not included in this appendix as they have been presented previously.

The variable names listed below are used in the Fortran source code of CONCORDT.FOR and in the corresponding flowcharts.

N1 -	The integer value that specifies the number of complex samples contained in the input sequence xn1().
N2 -	The integer value that specifies the number of complex samples contained in the input sequence xn2().
dsrce1 -	The character string 'F' or 'S' that specifies whether the input sequence xn1() is to be read from the input file (F) or generated (S) through use of the subroutine SAMPL1.

```
dsrce2 - The character string 'F' or 'S' that specifies
         whether the input sequence xn2() is to be read
         from the input file (F) or generated (S)
         through use of the subroutine SAMPL2.
option - The character string that specifies the
         operation to be performed as follows:
             'LCON' = Linear convolution,
             'LCOR' = Linear correlation,
             'CCON' = Circular convolution,
             'CCOR' = Circular correlation.
 xn1() - The first complex input sequence of length N1.
 xk1() - The sequence containing the DFT values of the
         array xn1(), i.e., xk1() = DFT[xn1()].
 xn2() - The second complex input sequence of length N2.
 xk2() - The sequence containing the DFT values of the
         array xn2(), i.e., xk2() = DFT[xn2()].
 xn3() - The complex output sequence.
 xk3() - The sequence containing the DFT values of the
         array xn3(), i.e., xk3() = DFT[xn3()].
```

Example #1

The circular convolution of the two sequences: xn1() = [1 3 5 7] and xn2() = [2 4 1 8] is demonstrated in this example. The listings that follow include the input file CONCORDT.IN and the tabular output file CONCORDT.OUT. The result of the circular convolution of these two sequences can be easily verified by manually performing the calculations. Manual calculation results in the sequence xn3() = [59 57 79 45]. This compares favorably with the computer generated output sequence.

CONCORDT.IN

```
004        F
004        F          CCON
1.0        0.0
3.0        0.0
5.0        0.0
7.0        0.0
2.0        0.0
4.0        0.0
1.0        0.0
8.0        0.0
```

CONCORDT.OUT

```
INPUT DATA SOURCEFILE: CONCORDT.IN
N1 =   4     dsrce1 = F
N2 =   4     dsrce2 = F
option = CCON


                INPUT DATA

                  xn1()
  n       REAL              IMAGINARY
  0     .100000E+01      .000000E+00
  1     .300000E+01      .000000E+00
  2     .500000E+01      .000000E+00
  3     .700000E+01      .000000E+00

                  xn2()
  n       REAL              IMAGINARY
  0     .200000E+01      .000000E+00
  1     .400000E+01      .000000E+00
  2     .100000E+01      .000000E+00
  3     .800000E+01      .000000E+00


               OUTPUT DATA

                  xn3()
  n       REAL              IMAGINARY
  0     .590000E+02      .190735E-05
  1     .570000E+02      .560272E-06
  2     .790000E+02     -.855914E-05
  3     .450000E+02      .551928E-05
```

Example #2

The next operation to be demonstrated is linear convolution. For this operation the two input sequences chosen were: xn1() = [1 2 3 4] and xn2() = [5 4 3 2 1]. As in the first example, the input sequences are short enough to check the solution via manual calculations. The listings that follow include both the input file CONCORDT.TST, as well as the output file CONCORDT.OUT. Manual calculation of the linear convolution results in the sequence: xn3() = [5 14 26 40 30 20 11 4]. The output produced by the program results in the same solution.

This example also appears in the header text of the program. The user can run this problem by selecting Test Mode and entering 'CONCORDT.TST' as the name of the input file.

CONCORDT.TST

```
004        F
005        F          LCON
1.0        0.0
2.0        0.0
3.0        0.0
4.0        0.0
5.0        0.0
4.0        0.0
3.0        0.0
2.0        0.0
1.0        0.0
```

CONCORDT.OUT

```
INPUT DATA SOURCEFILE: CONCORDT.TST
N1 =   4      dsrce1 = F
N2 =   5      dsrce2 = F
option = LCON


              INPUT DATA

                 xn1()
  n       REAL             IMAGINARY
  0     .100000E+01      .000000E+00
  1     .200000E+01      .000000E+00
  2     .300000E+01      .000000E+00
  3     .400000E+01      .000000E+00

                 xn2()
  n       REAL             IMAGINARY
  0     .500000E+01      .000000E+00
  1     .400000E+01      .000000E+00
  2     .300000E+01      .000000E+00
  3     .200000E+01      .000000E+00
  4     .100000E+01      .000000E+00


             OUTPUT DATA

                 xn3()
  n       REAL             IMAGINARY
  0     .500000E+01      .953*74E-06
  1     .140000E+02     -.303457E-05
  2     .260000E+02     -.756009E-05
  3     .400000E+02     -.404610E-05
  4     .300000E+02      .217716E-05
  5     .200000E+02      .762858E-05
  6     .110000E+02      .892130E-05
  7     .400001E+01      .472045E-05
```

Example #3

Using the same sequences that were used in Example #1, this example problem demonstrates the circular correlation operation. The input sequences are repeated here for ease of analysis: xn1() = [1 3 5 7] and xn2() = [2 4 1 8]. A listing of the tabular output file CONCORDT.OUT is included on the page that follows. The result of performing the calculations manually is the sequence xn3() = [75 61 63 41]. This compares favorably with the solution generated by the program.

CONCORDT.OUT

```
INPUT DATA SOURCEFILE: CONCORDT.IN
N1 =    4     dsrce1 = F
N2 =    4     dsrce2 = F
option = CCOR


              INPUT DATA


                xn1()
 n       REAL              IMAGINARY
 0      .100000E+01      .000000E+00
 1      .300000E+01      .000000E+00
 2      .500000E+01      .000000E+00
 3      .700000E+01      .000000E+00

                xn2()
 n       REAL              IMAGINARY
 0      .200000E+01      .000000E+00
 1      .400000E+01      .000000E+00
 2      .100000E+01      .000000E+00
 3      .800000E+01      .000000E+00



             OUTPUT DATA

                xn3()
 n       REAL              IMAGINARY
 0      .750000E+02      .000000E+00
 1      .610000E+02     -.108481E-05
 2      .630000E+02      .126364E-05
 3      .410000E+02     -.480426E-05
```

Example #4

This final example problem demonstrates the linear correlation computation. The sequence xn1() consists of 128 samples of the unit step function and the sequence xn2() consists of 128 samples of a square wave. The goal is to compute the linear correlation of the two sequences i.e., the sequence: $R_{xn1xn2}()$. Plots of the two input sequences, xn1() and xn2(), as well as the output sequence R() are provided on the pages that follow. Manual calculation of the solution to this problem is somewhat impractical. However, the plotted data allows the user to verify that the results are correct through graphical analysis. The wraparound phenomenon discussed in Chapter III is evident from the plot of the output sequence. The actual non-zero values for the linear correlation of these two input sequences, as they are defined in this problem, consists of the computed output sequence truncated at sample n = 127.

The linear correlation of two real sequences results in a sequence that is also real. Figure E.4 is a plot of the imaginary part of the output sequence for this example problem. These non-zero values result from the use of single-precision computations as part of the DFT algorithm. Since the user will experience similar results, the plot of the imaginary values is included in this analysis.

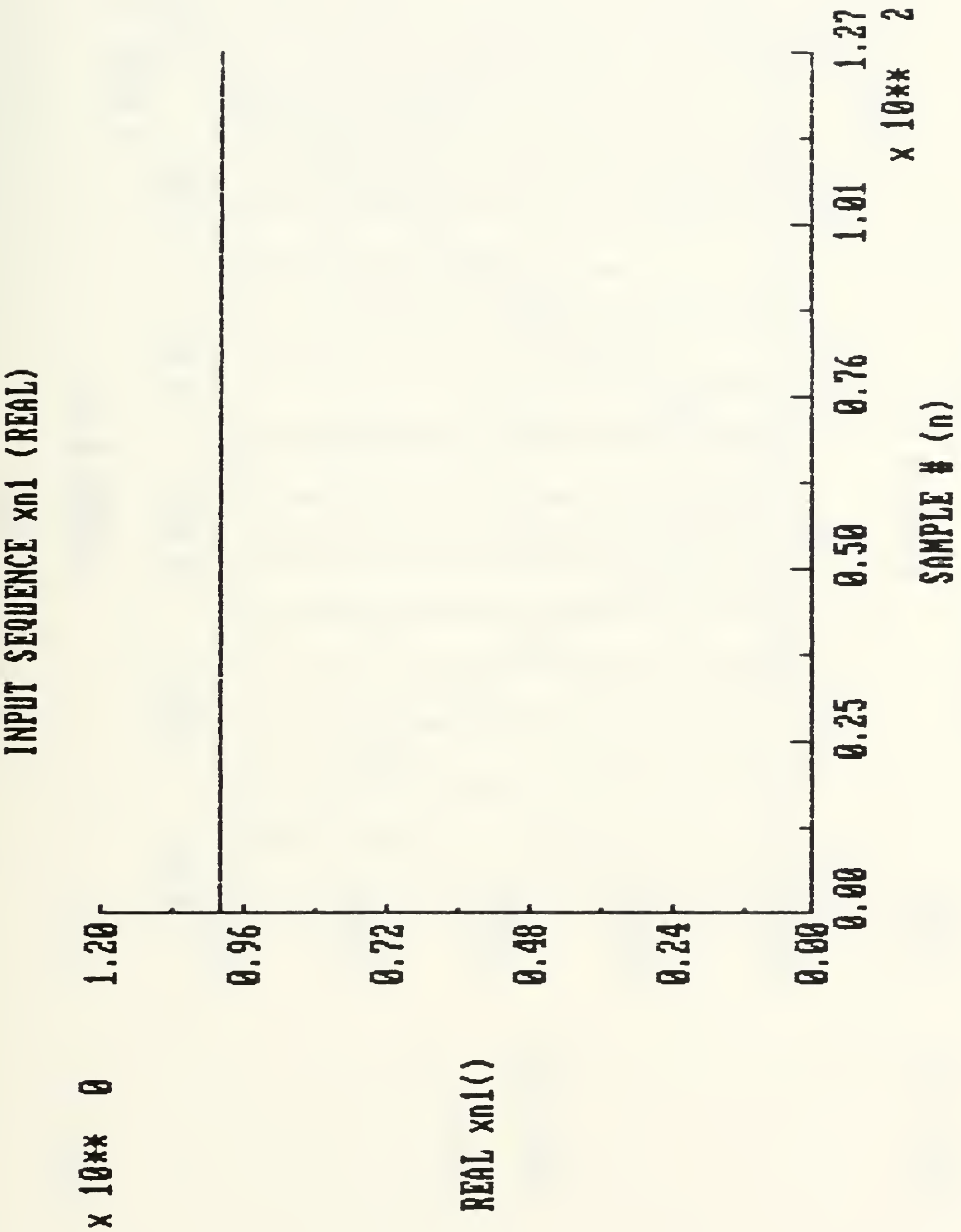


Figure E.1 Input sequence xn1(n) - Example #4.

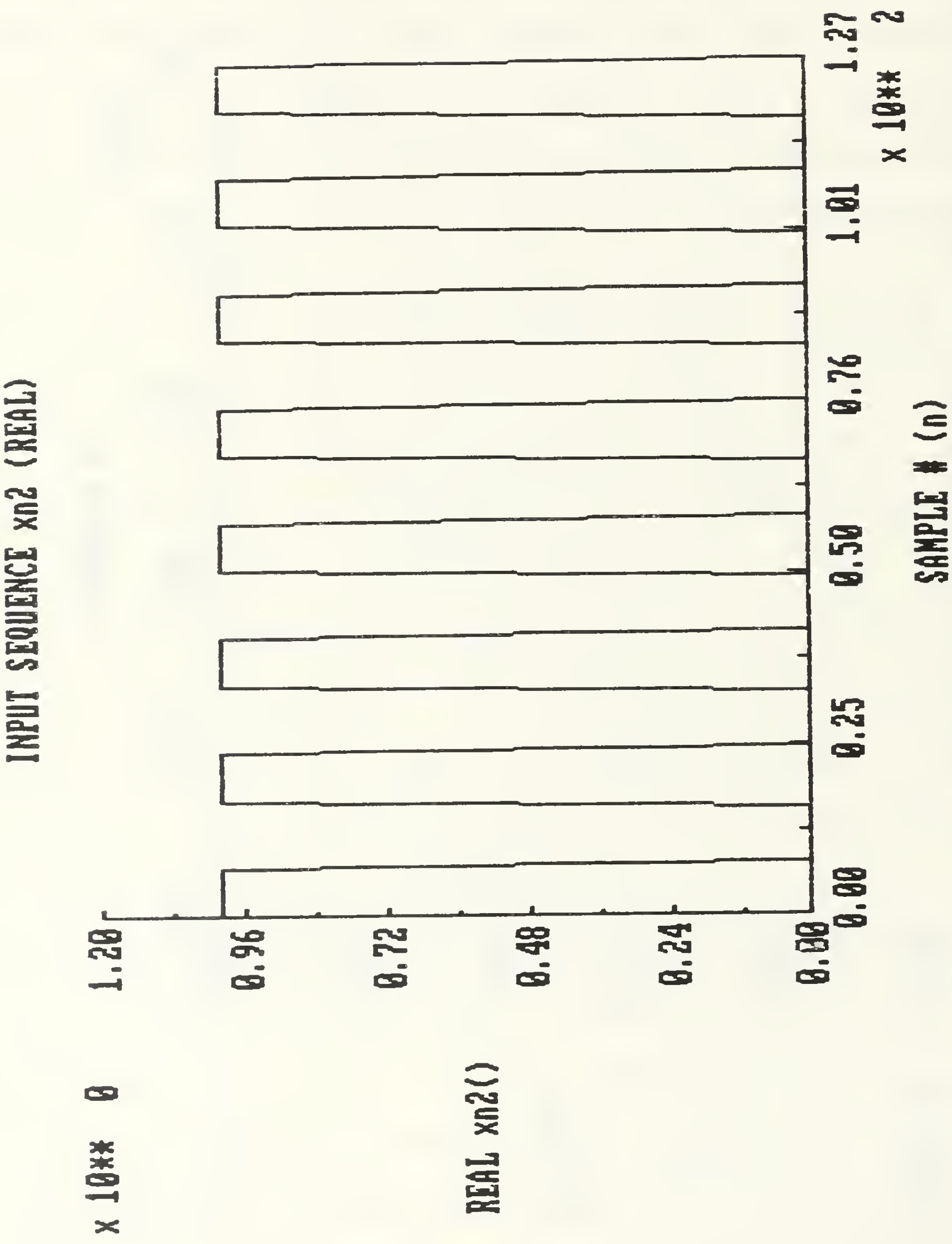


Figure E.2 Input sequence xn2(n) - Example #4.

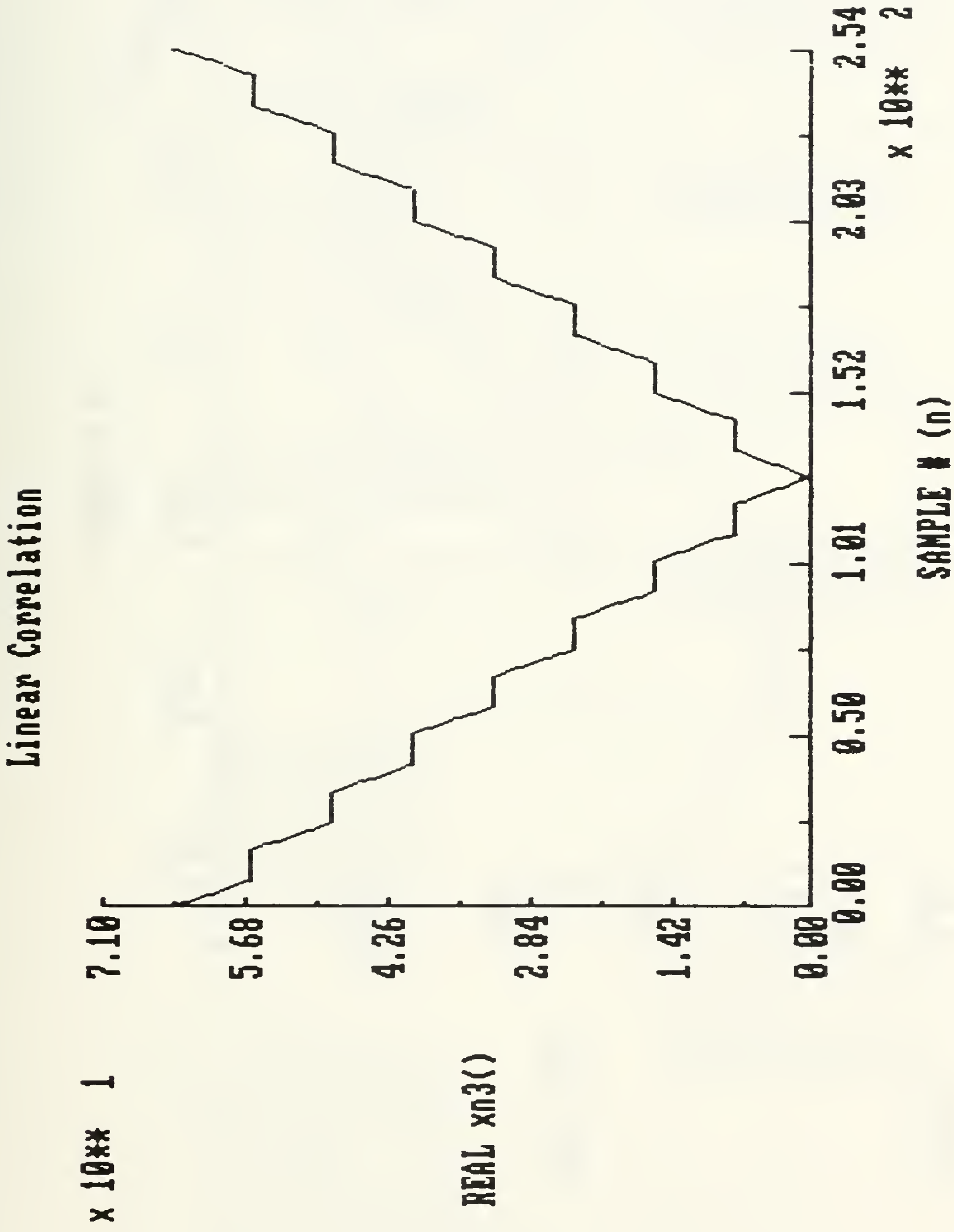


Figure E.3 The result of linear correlation (real part) - Example #4.

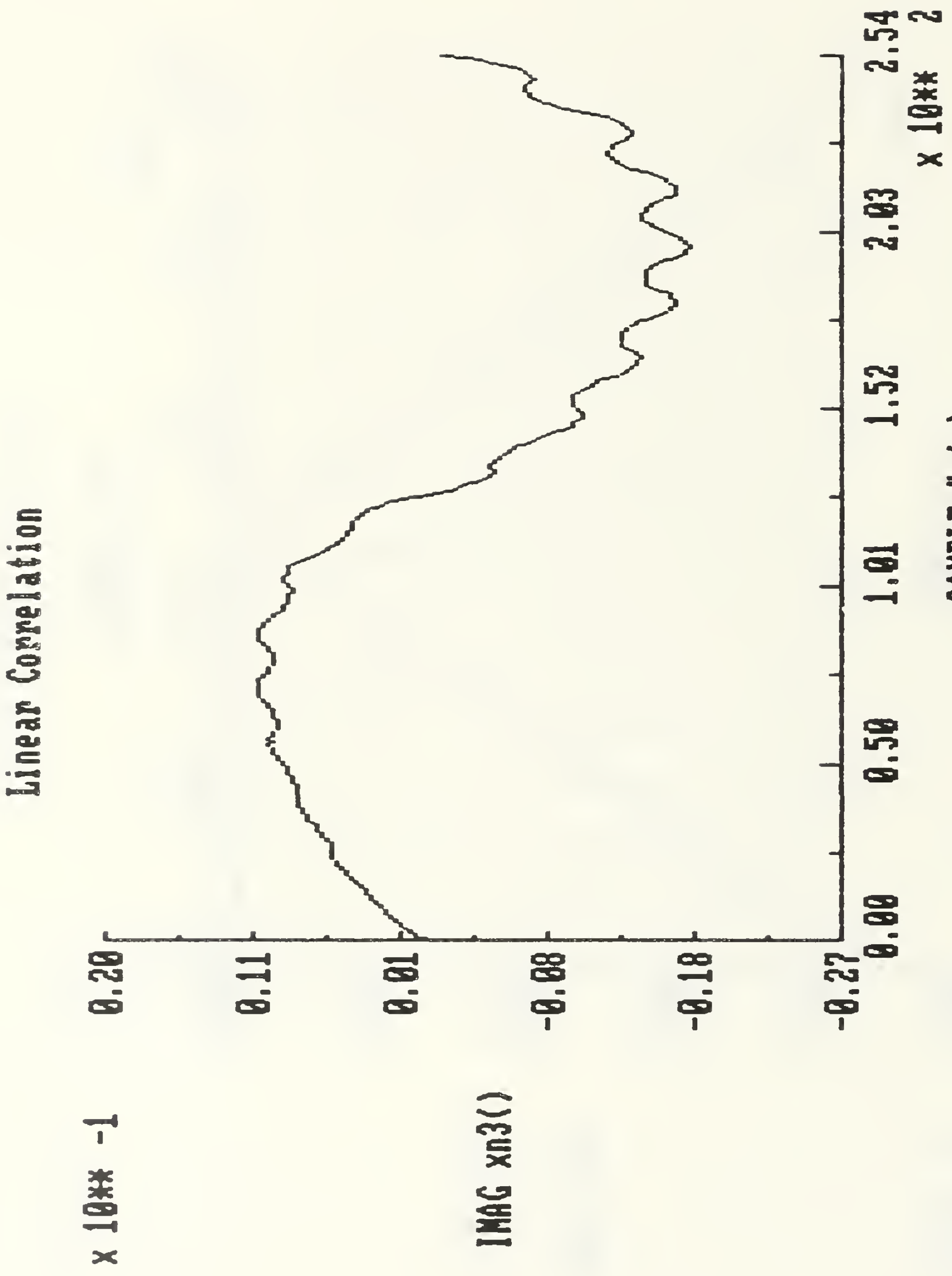


Figure E.4 The result of linear correlation (imaginary part) - Example #4.

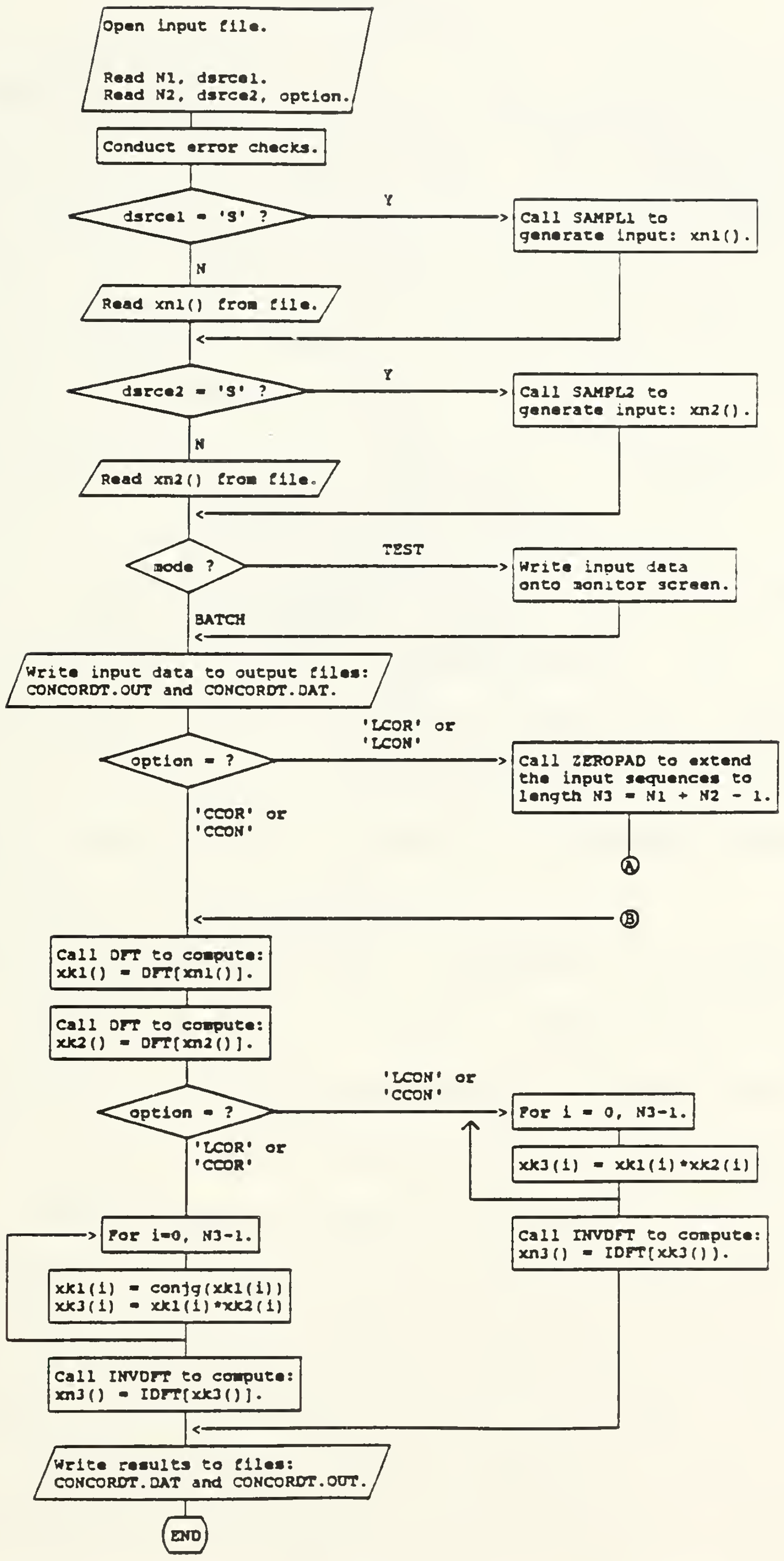

Open input file.
Read N1, dsrce1.
Read N2, dsrce2, option.
Conduct error checks.
dsrce1 = 'S' ?
Y
N
Call SAMPL1 to generate input: xn1().
Read xn1() from file.
dsrce2 = 'S' ?
Y
N
Call SAMPL2 to generate input: xn2().
Read xn2() from file.
mode ?
TEST
BATCH
Write input data onto monitor screen.
Write input data to output files: CONCORDT.OUT and CONCORDT.DAT.
option = ?
'LCOR' or 'LCON'
'CCOR' or 'CCON'
Call ZEROPAD to extend the input sequences to length N3 = N1 + N2 - 1.
A
B
Call DFT to compute: xk1() = DFT[xn1()].
Call DFT to compute: xk2() = DFT[xn2()].
option = ?
'LCON' or 'CCON'
'LCOR' or 'CCOR'
For i = 0, N3-1.
xk3(i) = xk1(i)*xk2(i)
Call INVDFT to compute: xn3() = IDFT[xk3()).
For i=0, N3-1.
xk1(i) = conjg(xk1(i))
xk3(i) = xk1(i)*xk2(i)
Call INVDFT to compute: xn3() = IDFT[xk3()].
Write results to files: CONCORDT.DAT and CONCORDT.OUT.
END

Figure E.5 CONCORDT.FOR Software Flowchart.

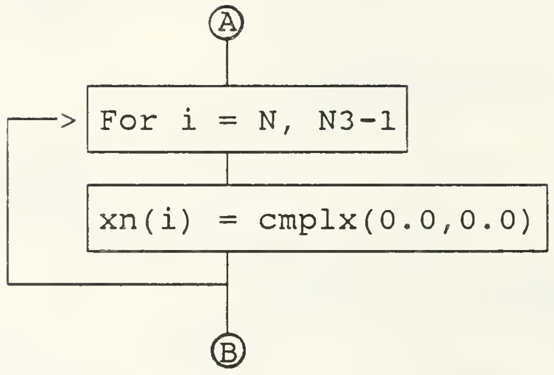


Figure E.6 ZEROPAD Subroutine Flowchart.

Appendix F

The program FFT.FOR computes either the FFT or the IFFT of a complex sequence of input data. This appendix contains an example problem that demonstrates each of these computations. The final pages of this appendix are the software flowcharts of the main program FFT.FOR and the subroutines FFT, INVFFT, and REVERSAL.

Development of the FFT algorithm is somewhat involved and in any case beyond the scope of this report. However, in order to make the software flowcharts and corresponding Fortran source code more understandable, the following synopsis pertains to the FFT algorithm as developed in Chapter 8 of Reference 1 and implemented by this program. The variable names used throughout correspond precisely to those presented in the reference.

The FFT computation for a sequence of length $N = 2^m$ values is broken up into m stages. Each stage consists of N/2 two-point DFT computations called 'butterflies'. In an effort to increase the computational efficiency of the algorithm, each butterfly occurring within a given stage and requiring use of the same weighting factor ($W = e^{-j2\pi r/N}$) is computed in a single loop, thus eliminating the requirement to recompute the weighting factor for each consecutive butterfly. The addresses (array indices) of the two values that participate in a butterfly computation are assigned the values: itop and ibot. The value corresponding to the separation between these indices is the value iwidth, i.e., iwidth = ibot - itop. The tradeoff of grouping the butterfly computations by their weighting factors is the determination of the correct participants for each butterfly in the group. The program determines the addresses of these participants through use of the values itop, ibot and iwidth. The efficiency gained by grouping the butterflies according to their weighting factors is a function of the number of values (N) comprising the input sequence.

The listing below further explains the function of the individual variables as they appear in the software.

m -	The integer value that specifies the number of complex samples contained in the input sequence, i.e., $N = 2^m$.
dsorce -	The character string 'F' or 'S' that specifies whether the input sequence xtmp() is to be read from the input file (F) or generated (S) through use of the subroutine SAMPLE.
option -	The character string 'FFT' or 'INV' that specifies the computation to be performed.
xtmp() -	The complex input sequence of length N.
x() -	The array containing the original input sequence but in bit-reversed order. After the subroutine FFT or INVFFT is called, this array contains the results of the FFT/IFFT computation in rectangular form, i.e., (real, imaginary).
xmag() -	The array containing the magnitude values of the output sequence.
xph() -	The array containing the phase (degrees) values of the output sequence.
L -	The integer value corresponding to the stage being computed.
iwidth -	The integer value corresponding to the address separation of the participants in a butterfly computation.
itop -	The integer value corresponding to the array index of the first participant in a butterfly computation.

ibot -	The integer value corresponding to the array index of the second participant in a butterfly computation.
ispace -	The integer value corresponding to the address separation between first participants in consecutive butterflies.
r -	The value corresponding to the index of the weighting factor.
W -	The complex weighting factor $W = e^{-j2\pi r/N}$ involved in each butterfly computation.
maddr -	The integer value corresponding to the original address of the elements of the input sequence.
newaddr -	The integer value corresponding to the new address assigned as a result of the bit-reversal algorithm.

Example #1

This example demonstrates both the FFT and the IFFT computations. The input sequence consists of N = 8 (m = 3) samples of the real sequence xtmp() = [0 1 2 3 4 0 0 0]. The imaginary part of each sample is assigned the value 0. Included on the page that follows are listings of the input file FFT.TST required to run this example problem, as well as the tabular output file FFT.OUT. In order to reproduce the original input sequence, the FFT results of the sequence were input to the program on a second run and the IFFT of this sequence was computed. Listings of the input and output files corresponding to this second run are also included.

This example problem is also developed in the header text of FFT.FOR and can be run by the user in Test Mode by using the data prestored in the input file FFT.TST.

FFT.TST

```
3           F           FFT
0.0         0.0
1.0         0.0
2.0         0.0
3.0         0.0
4.0         0.0
0.0         0.0
0.0         0.0
0.0         0.0
```

FFT.OUT

```
 INPUT DATA SOURCEFILE: FFT.TST
 VALUE OF m = 3     VALUE OF N (2**m) =    8
 dsorce = F     option = FFT

           INPUT DATA                                    INPUT DATA
                                                   (BIT-REVERSED ORDER)

SAMPLE #      REAL            IMAGINARY          REAL            IMAGINARY
   0        .000000E+00     .000000E+00     .000000E+00     .000000E+00
   1        .100000E+01     .000000E+00     .400000E+01     .000000E+00
   2        .200000E+01     .000000E+00     .200000E+01     .000000E+00
   3        .300000E+01     .000000E+00     .000000E+00     .000000E+00
   4        .400000E+01     .000000E+00     .100000E+01     .000000E+00
   5        .000000E+00     .000000E+00     .000000E+00     .000000E+00
   6        .000000E+00     .000000E+00     .300000E+01     .000000E+00
   7        .000000E+00     .000000E+00     .000000E+00     .000000E+00


                                  OUTPUT DATA

SAMPLE #      REAL            IMAGINARY        MAGNITUDE          PHASE
                                                                (DEGREES)
   0         .100000E+02     .000000E+00     .100000E+02     .000000E+00
   1        -.541421E+01    -.482843E+01     .725448E+01    -.138273E+03
   2         .200000E+01     .200000E+01     .282843E+01     .450000E+02
   3        -.258579E+01    -.828427E+00     .271525E+01    -.162236E+03
   4         .200000E+01     .000000E+00     .200000E+01     .000000E+00
   5        -.258579E+01     .828427E+00     .271525E+01     .162236E+03
   6         .200000E+01    -.200000E+01     .282843E+01    -.450000E+02
   7        -.541421E+01     .482843E+01     .725448E+01     .138273E+03
```

FFT.IN

```
3          F          INV
10.0       0.0
-5.41421   -4.82843
2.0        2.0
-2.58579   -.828427
2.0        0.0
-2.58579   .828427
2.0        -2.0
-5.41421   4.82843
```

FFT.OUT

```
INPUT DATA SOURCEFILE: FFT.IN
VALUE OF m = 3      VALUE OF N (2**m) =    8
dsorce = F     option = INV


            INPUT DATA                              INPUT DATA
                                               (BIT-REVERSED ORDER)

SAMPLE #     REAL          IMAGINARY         REAL          IMAGINARY
   0       .100000E+02    .000000E+00     .100000E+02    .000000E+00
   1      -.541421E+01   -.482843E+01     .200000E+01    .000000E+00
   2       .200000E+01    .200000E+01     .200000E+01    .200000E+01
   3      -.258579E+01   -.828427E+00     .200000E+01   -.200000E+01
   4       .200000E+01    .000000E+00    -.541421E+01   -.482843E+01
   5      -.258579E+01    .828427E+00    -.258579E+01    .828427E+00
   6       .200000E+01   -.200000E+01    -.258579E+01   -.828427E+00
   7      -.541421E+01    .482843E+01    -.541421E+01    .482843E+01


         OUTPUT DATA

SAMPLE #     REAL          IMAGINARY       MAGNITUDE         PHASE
                                                           (DEGREES)
   0       .000000E+00    .000000E+00     .000000E+00    .000000E+00
   1       .100000E+01   -.782270E-09     .100000E+01   -.448207E-07
   2       .200000E+01    .437114E-07     .200000E+01    .125224E-05
   3       .300000E+01    .218557E-07     .300000E+01    .417413E-06
   4       .400000E+01    .000000E+00     .400000E+01    .000000E+00
   5      -.172853E-05   -.429291E-07     .172907E-05   -.178577E+03
   6      -.834465E-06   -.437114E-07     .835609E-06   -.177001E+03
   7       .715256E-06    .218557E-07     .715590E-06    .175021E+01
```

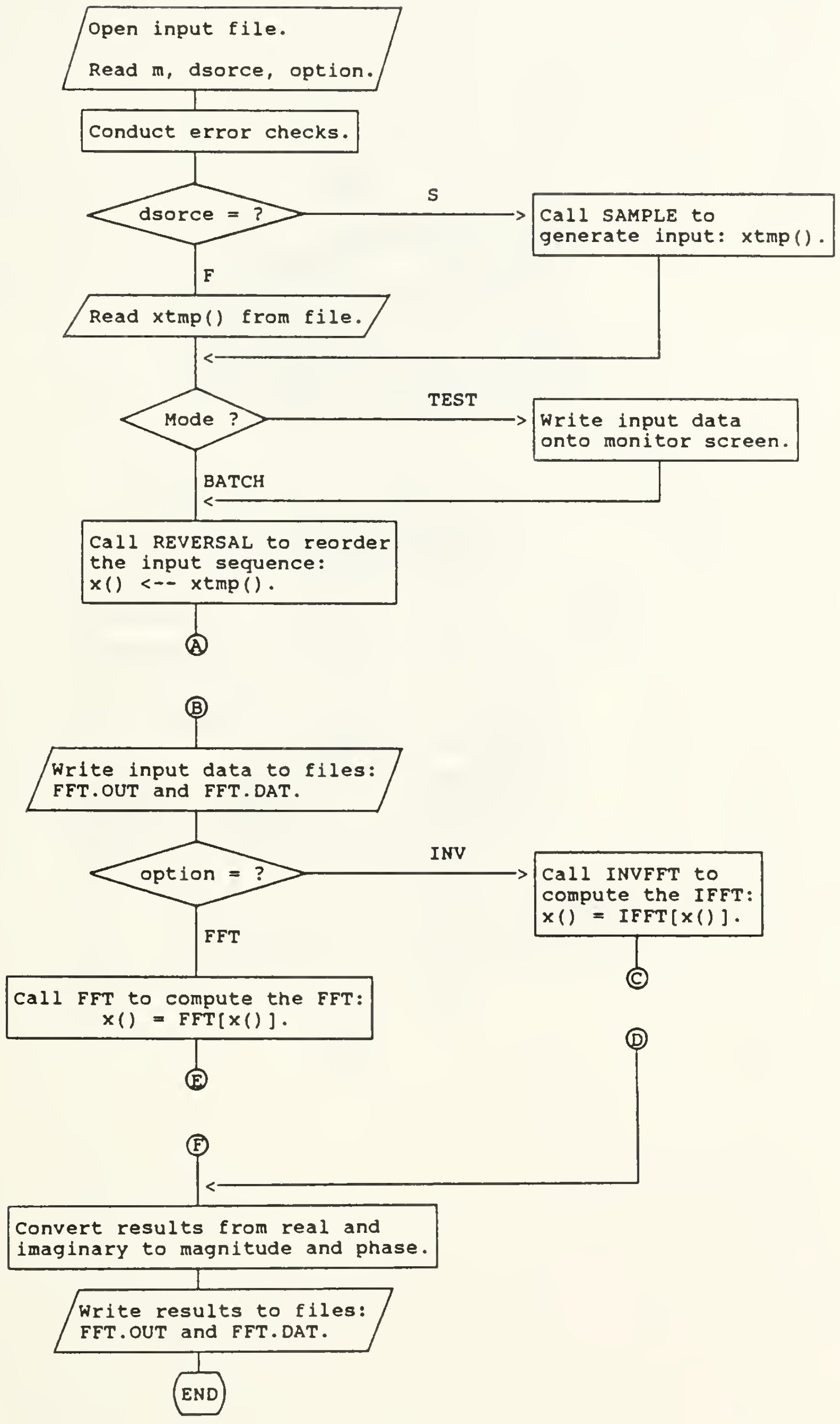

Open input file.
Read m, dsorce, option.
Conduct error checks.
dsorce = ?
S
Call SAMPLE to generate input: xtmp().
F
Read xtmp() from file.
Mode ?
TEST
Write input data onto monitor screen.
BATCH
Call REVERSAL to reorder the input sequence: x() <-- xtmp().
A
B
Write input data to files: FFT.OUT and FFT.DAT.
option = ?
INV
Call INVFFT to compute the IFFT: x() = IFFT[x()].
C
FFT
Call FFT to compute the FFT: x() = FFT[x()].
E
D
F
Convert results from real and imaginary to magnitude and phase.
Write results to files: FFT.OUT and FFT.DAT.
END

Figure F.1 FFT.FOR Software Flowchart.

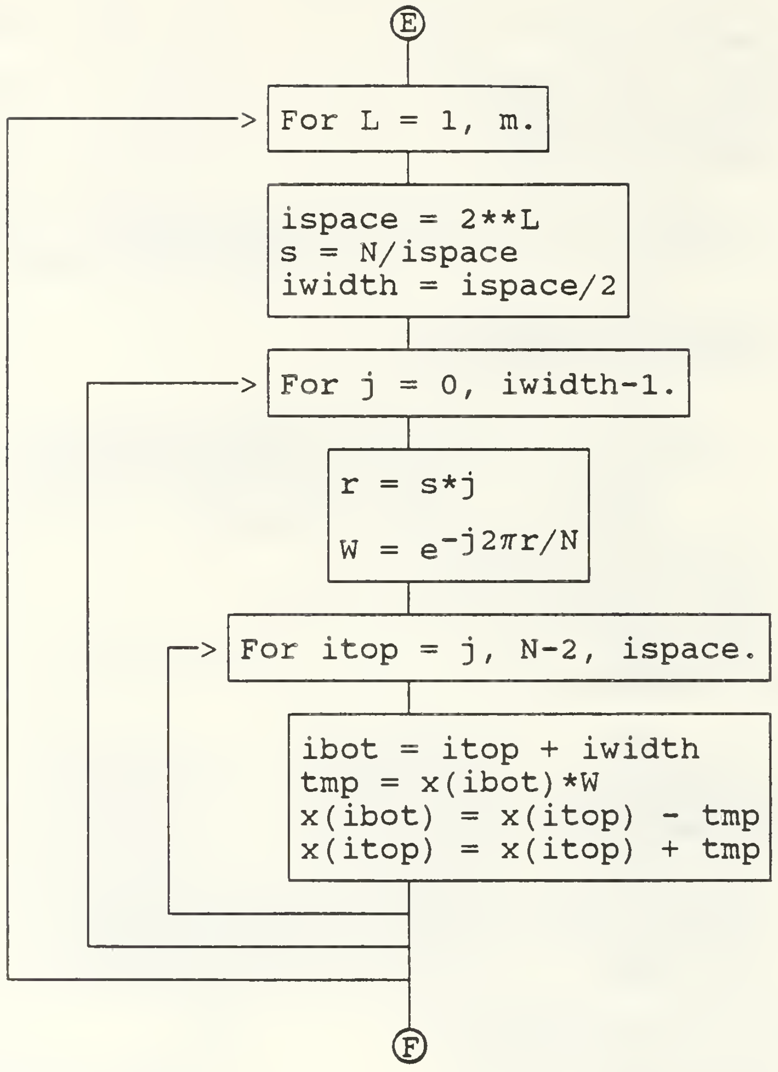


Figure F.2 FFT Subroutine Flow Chart.

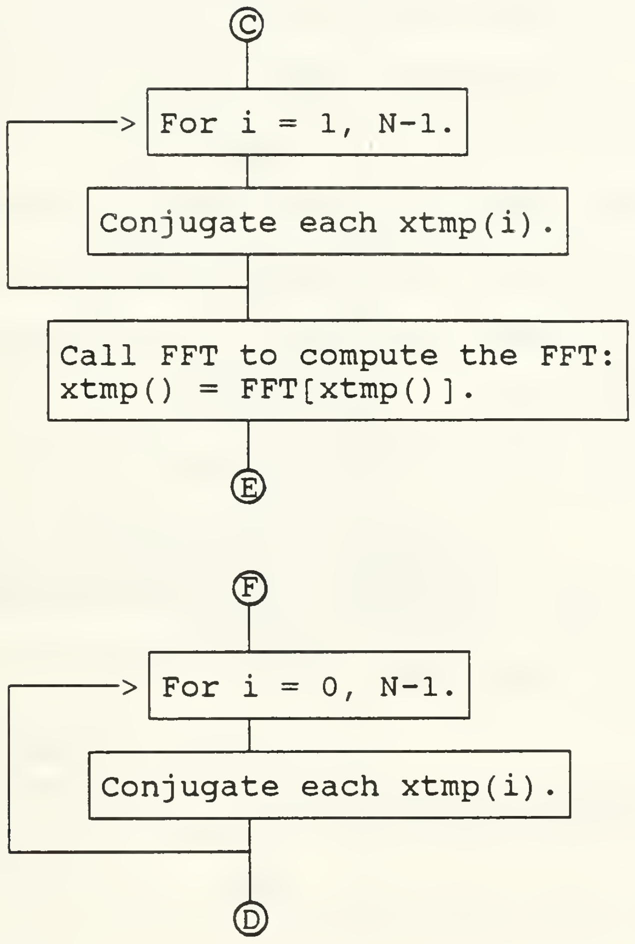


Figure F.3 INVFFT Subroutine Flowchart.

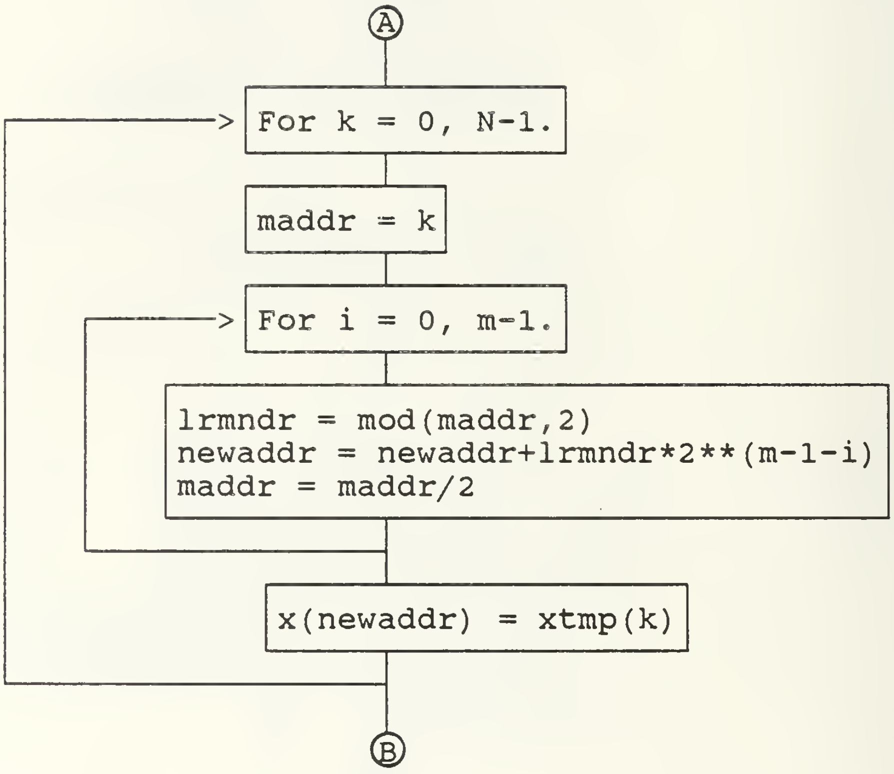


Figure F.4 REVERSAL Subroutine Flowchart.

Appendix G

The four computations that CONCORFT.FOR is capable of performing are demonstrated by the example problems included in this appendix. CONCORFT.FOR, like the other problem solving programs, generates an output file (CONCOR.DAT) that contains a listing of the input sequence(s), as well as the output sequence in a form suitable for plotting. Example #4 of this appendix includes plots of the input and output sequences used for that problem. The plots were produced by the program PLOTDAT.FOR. The final pages of this appendix are a flowchart of CONCORFT.FOR. Flowcharts of the six subroutines called by CONCORFT.FOR are not included in this appendix as each was presented previously.

The variable names listed below are used in the Fortran source code of CONCORFT.FOR and in the corresponding flowcharts.

N1 -	The integer value that specifies the number of complex samples contained in the input sequence xn1().
N2 -	The integer value that specifies the number of complex samples contained in the input sequence xn2().
dsrce1 -	The character string 'F' or 'S' that specifies whether the input sequence xn1() is to be read from the input file (F) or generated (S) through use of the subroutine SAMPL1.
dsrce2 -	The character string 'F' or 'S' that specifies whether the input sequence xn2() is to be read from the input file (F) or generated (S) through use of the subroutine SAMPL2.
option -	The character string that specifies the operation to be performed as follows: 'LCON' = Linear convolution, 'LCOR' = Linear correlation,

```
                 'CCON' = Circular convolution,
                 'CCOR' = Circular correlation.
   xn1() - The first complex input sequence of length N1.
 xtmp1() - A dummy array used for computations involving
           the array xn1().
   xn2() - The second complex input sequence of length N2.
 xtmp2() - A dummy array used for computations involving
           the array xn2().
   xn3() - The complex output sequence.
 xtmp3() - A dummy array used for computations involving
           the array xn3().
```

Example #1

This example demonstrates the circular convolution operation. The input sequences consist of the following real values: xn1() = [1 3 5 7] and xn2() = [2 4 1 8]. The result of manually calculating the circular convolution of these two sequences is the sequence: xn3() = [59 57 79 45]. The listings that follow include the input file CONCORFT.IN, required to run this problem, and the tabular output file CONCORFT.OUT containing the computed results. As can be seen from the listing of CONCORFT.OUT, the computation produced the anticipated results.

CONCORFT.IN

```
004         F
004         F           CCON
1.0         0.0
3.0         0.0
5.0         0.0
7.0         0.0
2.0         0.0
4.0         0.0
1.0         0.0
8.0         0.0
```

CONCORFT.OUT

```
 INPUT DATA SOURCEFILE: CONCORFT.IN
 N1 =    4     dsrce1 = F           N2 =    4      dsrce2 = F
option = CCON


                INPUT DATA

                  xn1()
     n      REAL              IMAGINARY
     0     .100000E+01     .000000E+00
     1     .300000E+01     .000000E+00
     2     .500000E+01     .000000E+00
     3     .700000E+01     .000000E+00

                  xn2()
     n      REAL              IMAGINARY
     0     .200000E+01     .000000E+00
     1     .400000E+01     .000000E+00
     2     .100000E+01     .000000E+00
     3     .800000E+01     .000000E+00


               OUTPUT DATA

                  xn3()
     n      REAL              IMAGINARY
     0     .590000E+02     .715256E-06
     1     .570000E+02     .262268E-06
     2     .790000E+02    -.715256E-06
     3     .450000E+02    -.262268E-06
```

Example #2

This example problem is developed in the header text of CONCORFT.FOR and can be run by the user by selecting Test Mode and using the input data prestored in the file CONCORFT.TST. The goal of this example is to compute the linear convolution of the two sequences: xn1() = [1 1 1 1] and xn2() = [2 2 2 2 2]. The sequence that should result from the operation is: xn3() = [2 4 6 8 8 6 4 2]. A listing of the input file CONCORFT.TST required to run this problem appears below.

CONCORFT.TST

```
004        F
005        F           LCON
1.0        0.0
1.0        0.0
1.0        0.0
1.0        0.0
2.0        0.0
2.0        0.0
2.0        0.0
2.0        0.0
2.0        0.0
```

A listing of the tabular output file CONCORFT.OUT is included on the page that follows. The computed output sequence compares favorably with the anticipated result.

CONCORFT.OUT

```
INPUT DATA SOURCEFILE: CONCORFT.TST
N1 =    4      dsrce1 = F            N2 =    5        dsrce2 = F
option = LCON
                 INPUT DATA

                   xn1()
     n       REAL                IMAGINARY
     0     .100000E+01      .000000E+00
     1     .100000E+01      .000000E+00
     2     .100000E+01      .000000E+00
     3     .100000E+01      .000000E+00

                   xn2()
     n       REAL                IMAGINARY
     0     .200000E+01      .000000E+00
     1     .200000E+01      .000000E+00
     2     .200000E+01      .000000E+00
     3     .200000E+01      .000000E+00
     4     .200000E+01      .000000E+00


                OUTPUT DATA

                   xn3()
     n       REAL                IMAGINARY
     0     .200000E+01      .894070E-07
     1     .400000E+01     -.421468E-07
     2     .600000E+01     -.754979E-07
     3     .800000E+01     -.168587E-06
     4     .800000E+01     -.894070E-07
     5     .600000E+01      .421468E-07
     6     .400000E+01      .754979E-07
     7     .200000E+01      .168587E-06
```

Example #3

Using the same input sequences as Example #1, the circular correlation operation is demonstrated by this problem. The input sequences are: xn1() = [1 3 5 7] and xn2() = [2 4 1 8]. For this computation, the anticipated result is the sequence: xn3() = [75 61 63 41]. A listing of the tabular output file is included below. The computed output sequence, xn3(), compares favorably with the anticipated result.

CONCORFT.OUT

```
 INPUT DATA SOURCEFILE: CONCORFT.IN
 N1 =    4      dsrce1 = F            N2 =    4      dsrce2 = F
option = CCOR


                 INPUT DATA

                   xn1()
    n      REAL            IMAGINARY
    0     .100000E+01     .000000E+00
    1     .300000E+01     .000000E+00
    2     .500000E+01     .000000E+00
    3     .700000E+01     .000000E+00

                   xn2()
    n      REAL            IMAGINARY
    0     .200000E+01     .000000E+00
    1     .400000E+01     .000000E+00
    2     .100000E+01     .000000E+00
    3     .800000E+01     .000000E+00


                OUTPUT DATA

                   xn3()
    n      REAL            IMAGINARY
    0     .750000E+02     .000000E+00
    1     .610000E+02     .198695E-06
    2     .630000E+02     .000000E+00
    3     .410000E+02    -.198695E-06
```

Example #4

This example problem demonstrates the linear correlation computation. The sequence xn1() consists of 128 samples of the unit step function and the sequence xn2() consists of 128 samples of a square wave. The goal is to compute the linear correlation of the two sequences, i.e., the sequence: $R_{xn1xn2}()$. Plots of the two input sequences, xn1() and xn2(), as well as the output sequence R() are provided on the pages that follow. As discussed previously (Example #4 of Appendix E), wraparound of the output sequence is produced by the program due to the zero padding required by use of the DFT technique. The wraparound results in non-zero values of the output sequence in the interval: 128 to 254. The plot of the output sequence clearly shows the wraparound phenomenon.

Example #4 of Appendix E also discussed the non-zero imaginary output values that are produced by the program CONCORDT.FOR when correlating two real input sequences. As exhibited by Figure G.4, CONCORFT.FOR produces similar results. The non-zero values are attributed to the single precision FFT algorithm used in the correlation computation.

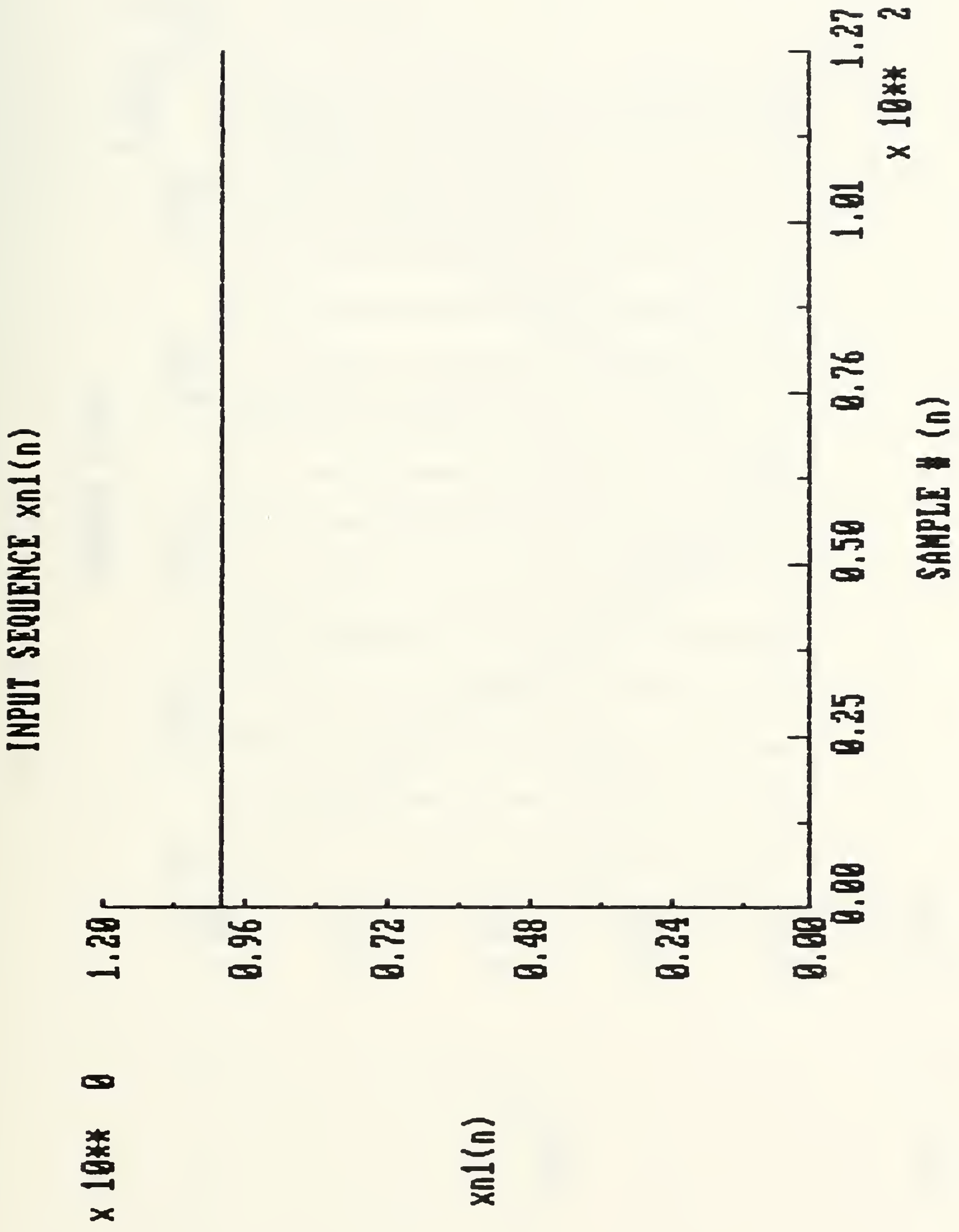


Figure G.1 Input sequence xn1(n) - Example #4.

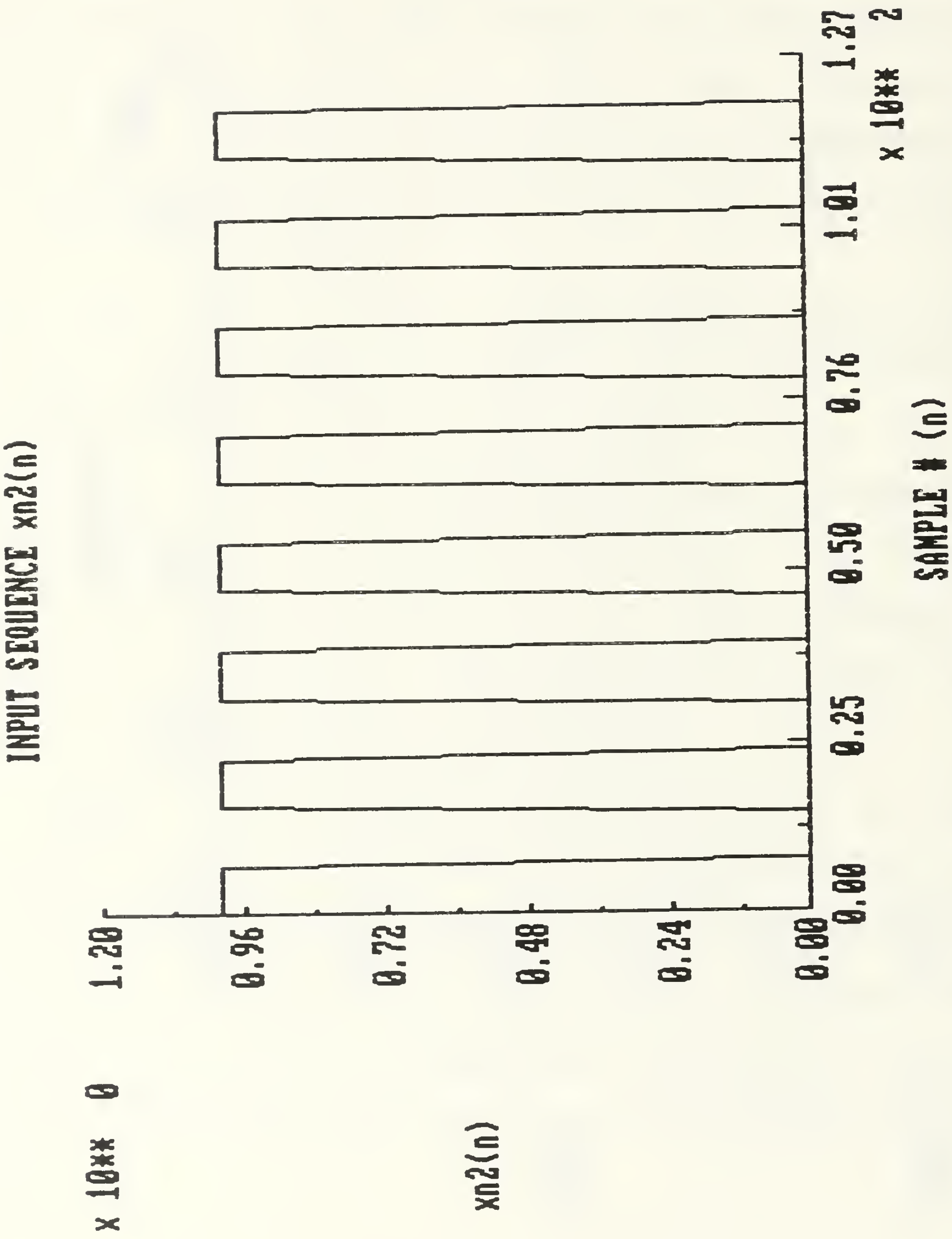


Figure G.2 Input sequence xn2(n) - Example #4.

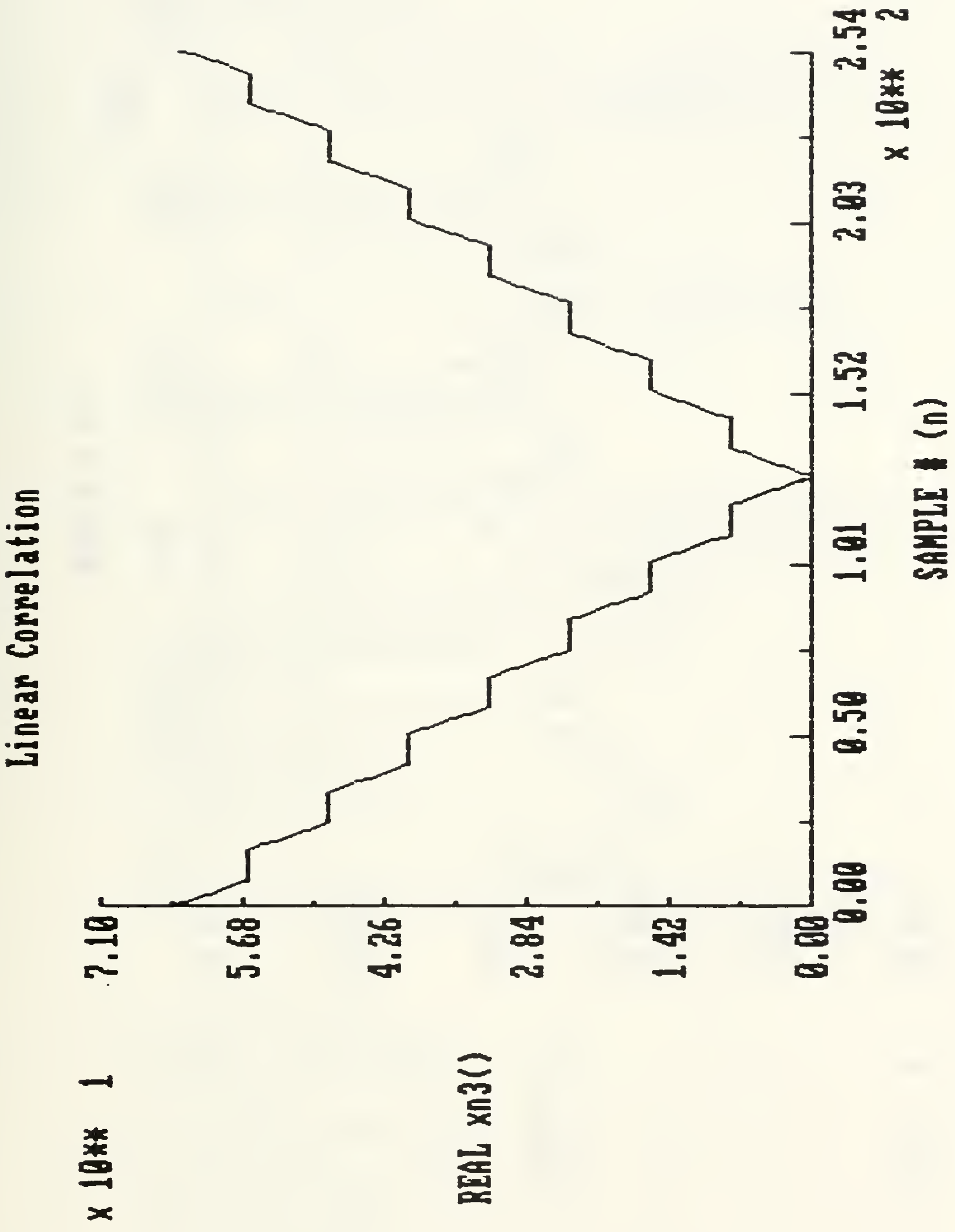


Figure G.3 The results of linear correlation (real part) - Example #4.

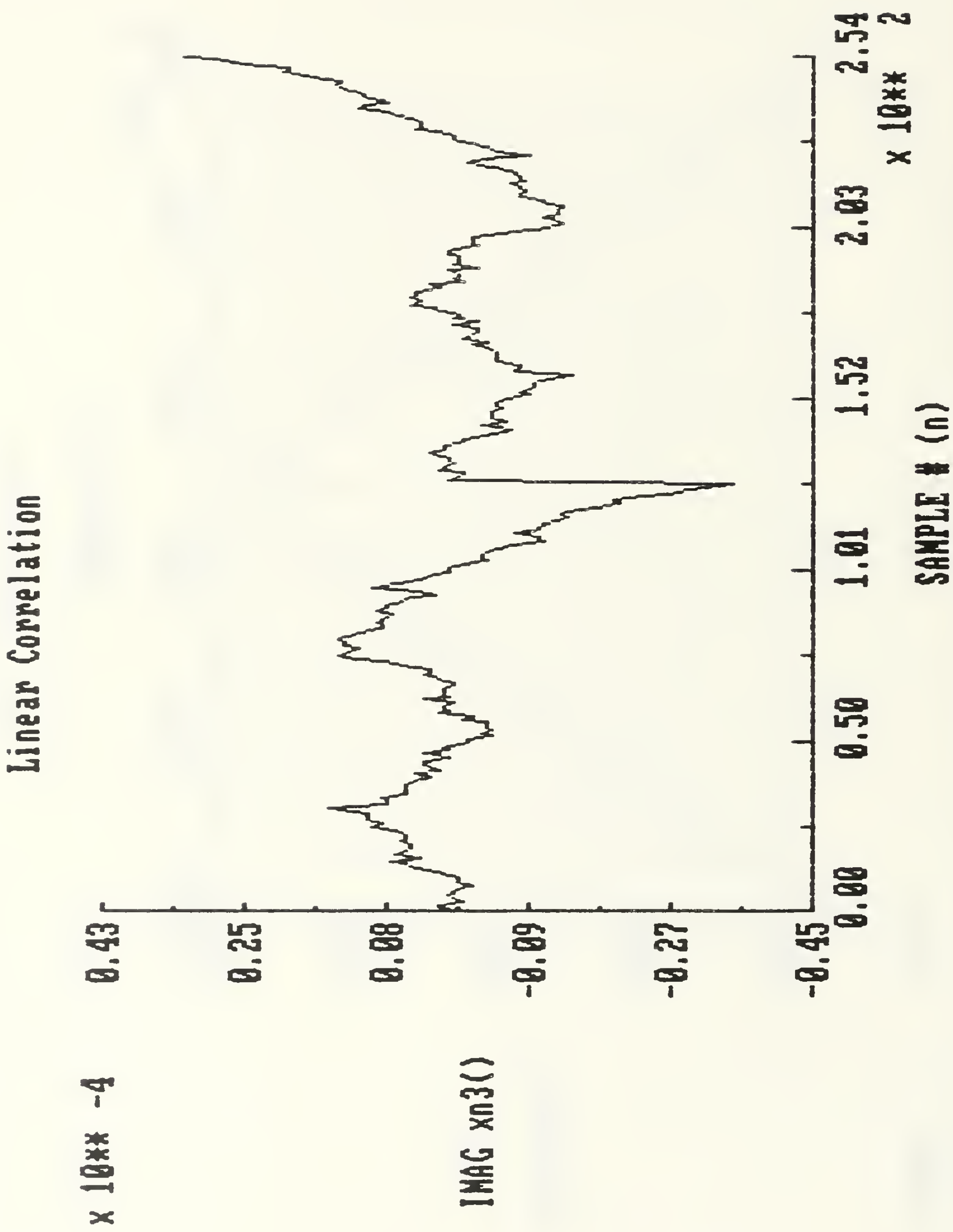
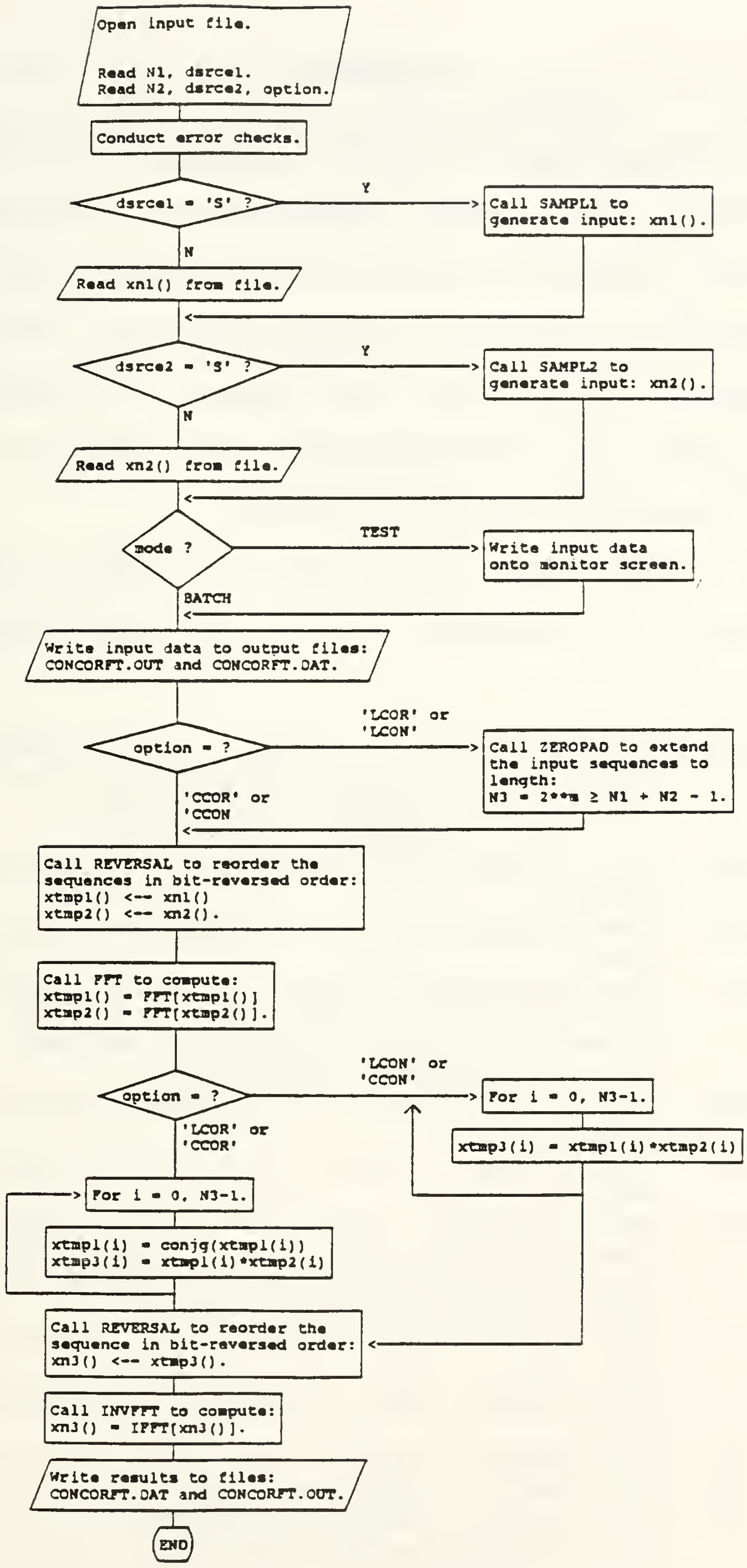


Figure G.4 The results of linear correlation (imaginary part) - Example #4.



Open input file.
Read N1, dsrce1.
Read N2, dsrce2, option.
Conduct error checks.
dsrce1 = 'S' ?
Y
Call SAMPL1 to generate input: xn1().
N
Read xn1() from file.
dsrce2 = 'S' ?
Y
Call SAMPL2 to generate input: xn2().
N
Read xn2() from file.
mode ?
TEST
Write input data onto monitor screen.
BATCH
Write input data to output files: CONCORFT.OUT and CONCORFT.DAT.
option = ?
'LCOR' or 'LCON'
Call ZEROPAD to extend the input sequences to length: N3 = 2**m ≥ N1 + N2 - 1.
'CCOR' or 'CCON
Call REVERSAL to reorder the sequences in bit-reversed order: xtmp1() <-- xn1() xtmp2() <-- xn2().
Call FFT to compute: xtmp1() = FFT[xtmp1()] xtmp2() = FFT[xtmp2()].
option = ?
'LCON' or 'CCON'
For i = 0, N3-1.
xtmp3(i) = xtmp1(i)*xtmp2(i)
'LCOR' or 'CCOR'
For i = 0, N3-1.
xtmp1(i) = conjg(xtmp1(i))
xtmp3(i) = xtmp1(i)*xtmp2(i)
Call REVERSAL to reorder the sequence in bit-reversed order: xn3() <-- xtmp3().
Call INVFFT to compute: xn3() = IFFT[xn3()].
Write results to files: CONCORFT.DAT and CONCORFT.OUT.
END

Figure G.5 CONCORFT.FOR Software Flowchart.

Appendix H

The program CONCOR.FOR will compute either the linear convolution or the linear correlation of two sequences of input data, depending on the option selected by the user. This appendix includes two example problems, each of which demonstrates one of these computations. The last pages of this appendix are the flowcharts that describe CONCOR.FOR and the subroutines CONVOL and CORREL.

The variable names listed below are used in the Fortran source code of CONCOR.FOR and in the corresponding flowcharts.

option -	The character string that specifies the operation to be performed as follows: 'LCON' = Linear convolution, 'LCOR' = Linear correlation.
ns1 -	The integer value denoting the starting point of xn1().
ne1 -	The integer value denoting the ending point of xn1().
dsrce1 -	The character string 'F' or 'S' that specifies whether the input sequence xn1() is to be read from the input file (F) or generated (S) through use of the subroutine SAMPL1.
ns2 -	The integer value denoting the starting point of xn2().
ne2 -	The integer value denoting the ending point of xn2().
dsrce2 -	The character string 'F' or 'S' that specifies whether the input sequence xn2() is to be read from the input file (F) or generated (S) through use of the subroutine SAMPL2.
xn1() -	The first input sequence of length N1 = ne1-ns1 + 1.
xn2() -	The second input sequence of length N2 = ne2-ns2 + 1.
yn() -	The output sequence of length N3 = N1 + N2 - 1 produced if option = 'LCON'.
ns3 -	The integer value corresponding to the starting point of the output sequence.

ne3 - The integer value corresponding to the ending point of the output sequence.

R() - The output sequence of length N3 = ne3 - ns3 + 1 produced if option = 'LCOR'.

Example #1

The first computation to be demonstrated is the linear convolution of the two sequences: xn1(n) = [1 1 1 1] for -3 ≤ n ≤ 0 (ns1 = -3, ne1 = 0) and xn2(n) = [1 2 3 4 5] for 0 ≤ n ≤ 4 (ns2 = 0, ne2 = 4). To run this example problem the input file CONCOR.TST was created. A listing of this file appears below.

CONCOR.TST

```
LCON
  -3      0000      F
0000      0004      F
1.0
1.0
1.0
1.0
1.0
2.0
3.0
4.0
5.0
```

This example is also developed in the header text of CONCOR.FOR and can be run by the user in Test Mode by specifying the input file CONCOR.TST. The computed output, as it appears in the file CONCOR.OUT, is included on the page that follows. In addition to the tabulated data, plots of the input and output sequences are also included. The plotting program PLOTDAT.FOR will not attempt to connect the plotted values of sequences consisting of less than 25 points. Instead, the symbol '+' is placed on the plot at the appropriate locations. This example problem was chosen to demonstrate this feature of PLOTDAT.FOR.

The result of manually calculating the linear convolution of the two input sequences is the sequence: yn() = [1 3 6 10 14 12 9 5]. This compares favorably with the computer generated results.

<u>CONCOR.OUT</u>

```
 INPUT DATA SOURCEFILE: CONCOR.TST
 ns1 =   -3   ne1 =    0   dsrce1 = F
 ns2 =    0   ne2 =    4   dsrce2 = F
 option = LCON


      INPUT DATA

       n      xn1(n)

      -3     .100000E+01
      -2     .100000E+01
      -1     .100000E+01
       0     .100000E+01

       n      xn2(n)

       0     .100000E+01
       1     .200000E+01
       2     .300000E+01
       3     .400000E+01
       4     .500000E+01


      OUTPUT DATA

       n       yn(n)

      -3     .100000E+01
      -2     .300000E+01
      -1     .600000E+01
       0     .100000E+02
       1     .140000E+02
       2     .120000E+02
       3     .900000E+01
       4     .500000E+01
```

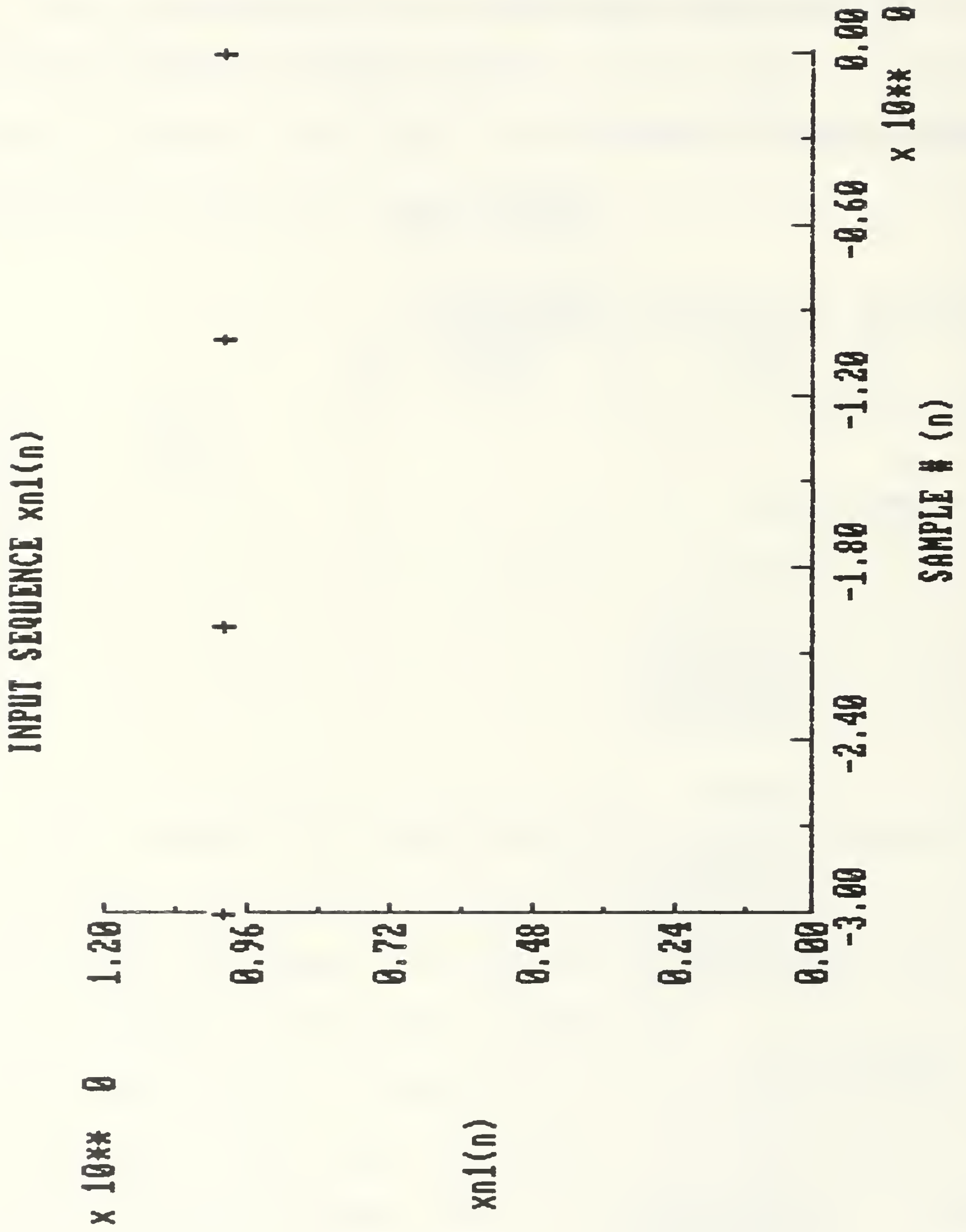


Figure H.1 Input sequence xn1(n) - Example #1.

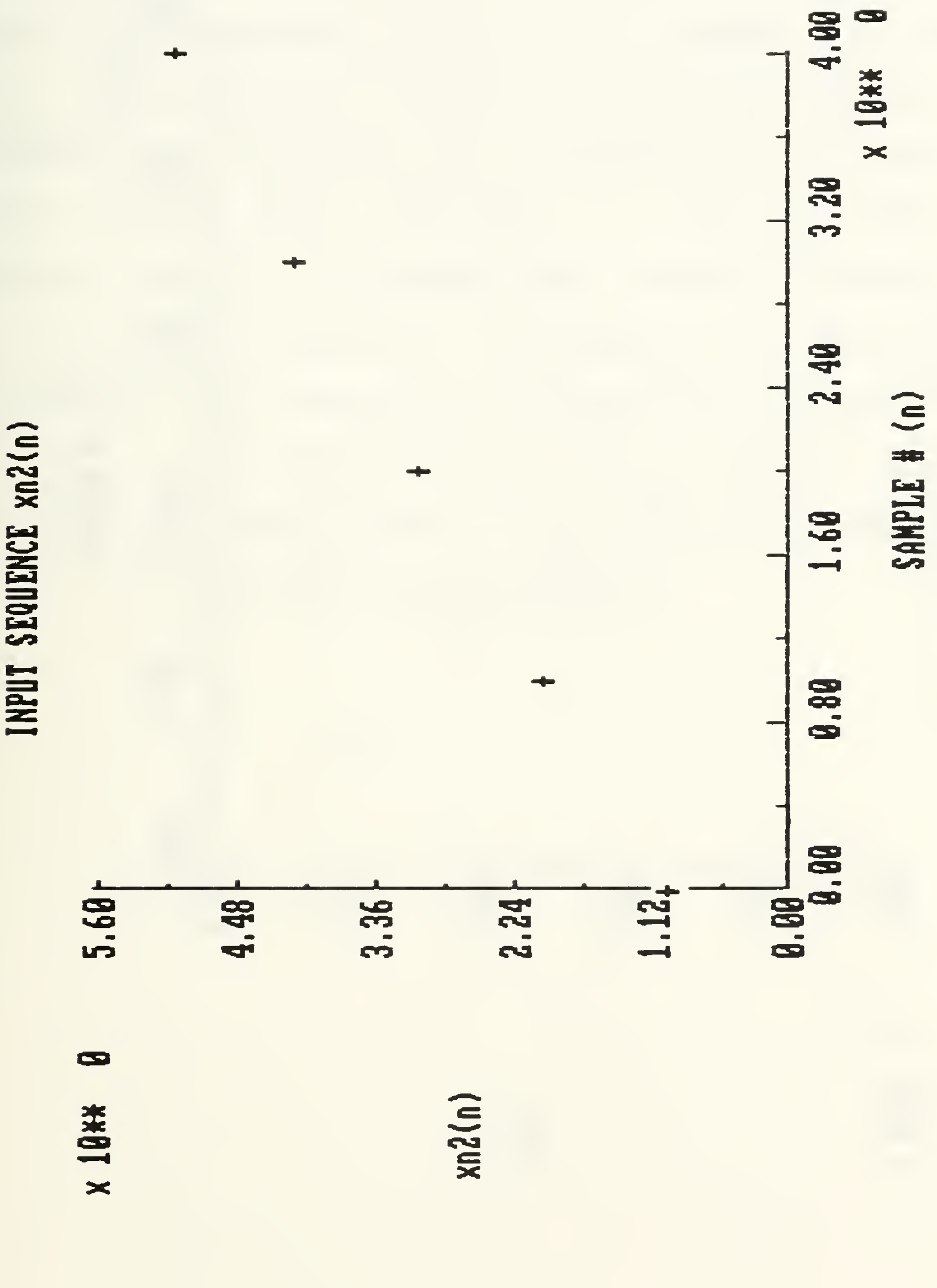


Figure H.2 Input sequence xn2(n) - Example #1.

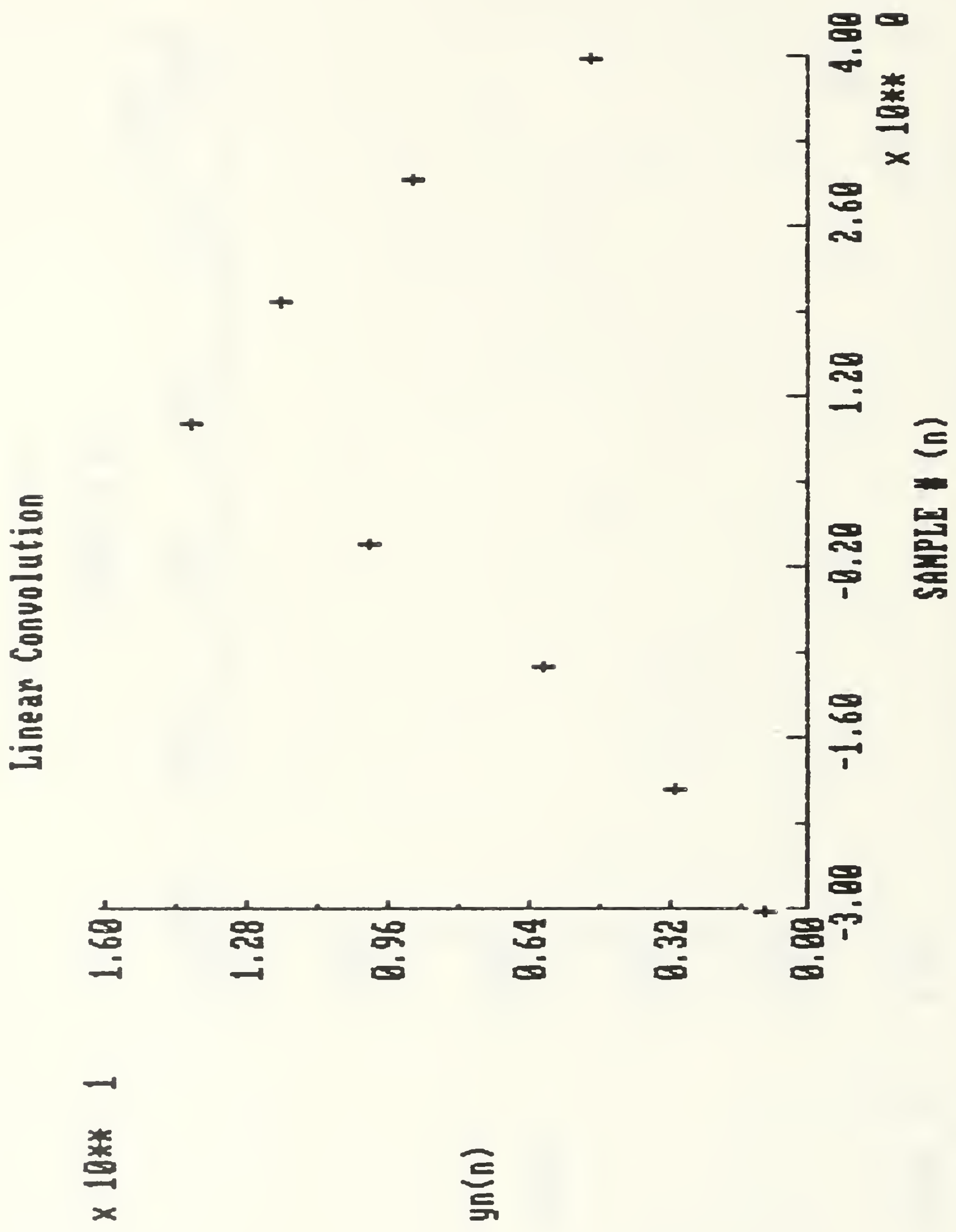


Figure H.3 The result of linear convolution - Example #1.

Example #2

This example demonstrates the linear correlation option. The input sequences chosen are identical to those used in Example #4 of Appendix E. The input sequence xn1() consists of N1 = 128 values of the unit step function, and the input sequence xn2() consists of N2 = 128 values of a square wave. Plots of the input sequences, as well as the output sequence, appear on the pages that follow. The results of the correlation operation, as produced by CONCOR.FOR, are similar to those produced by CONCORDT.FOR and CONCORFT.FOR. However, as the plots indicate, the wraparound phenomenon exhibited previously does not occur when the sequences are linearly correlated in the time domain.

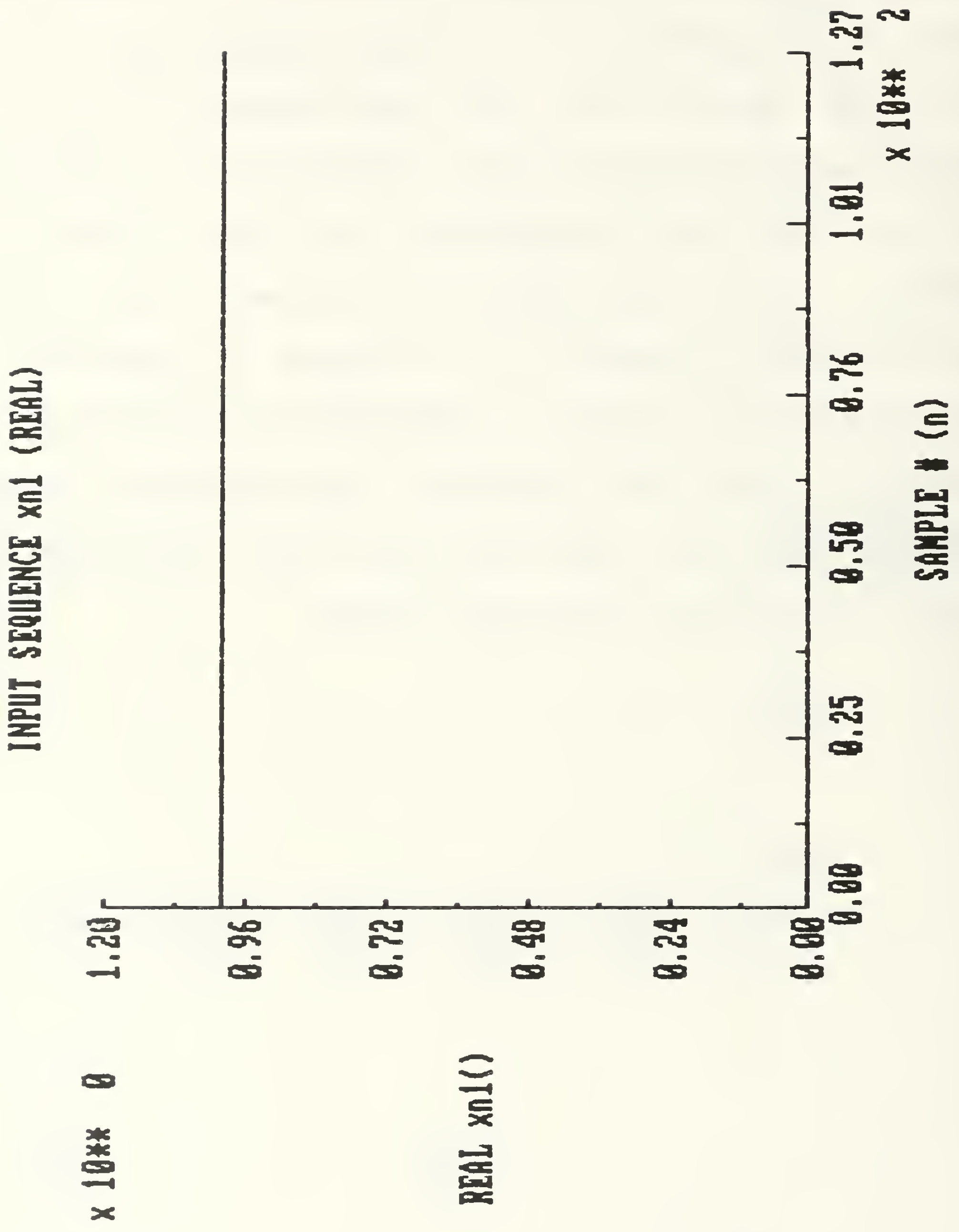


Figure H.4 Input sequence xn1(n) - Example #2.

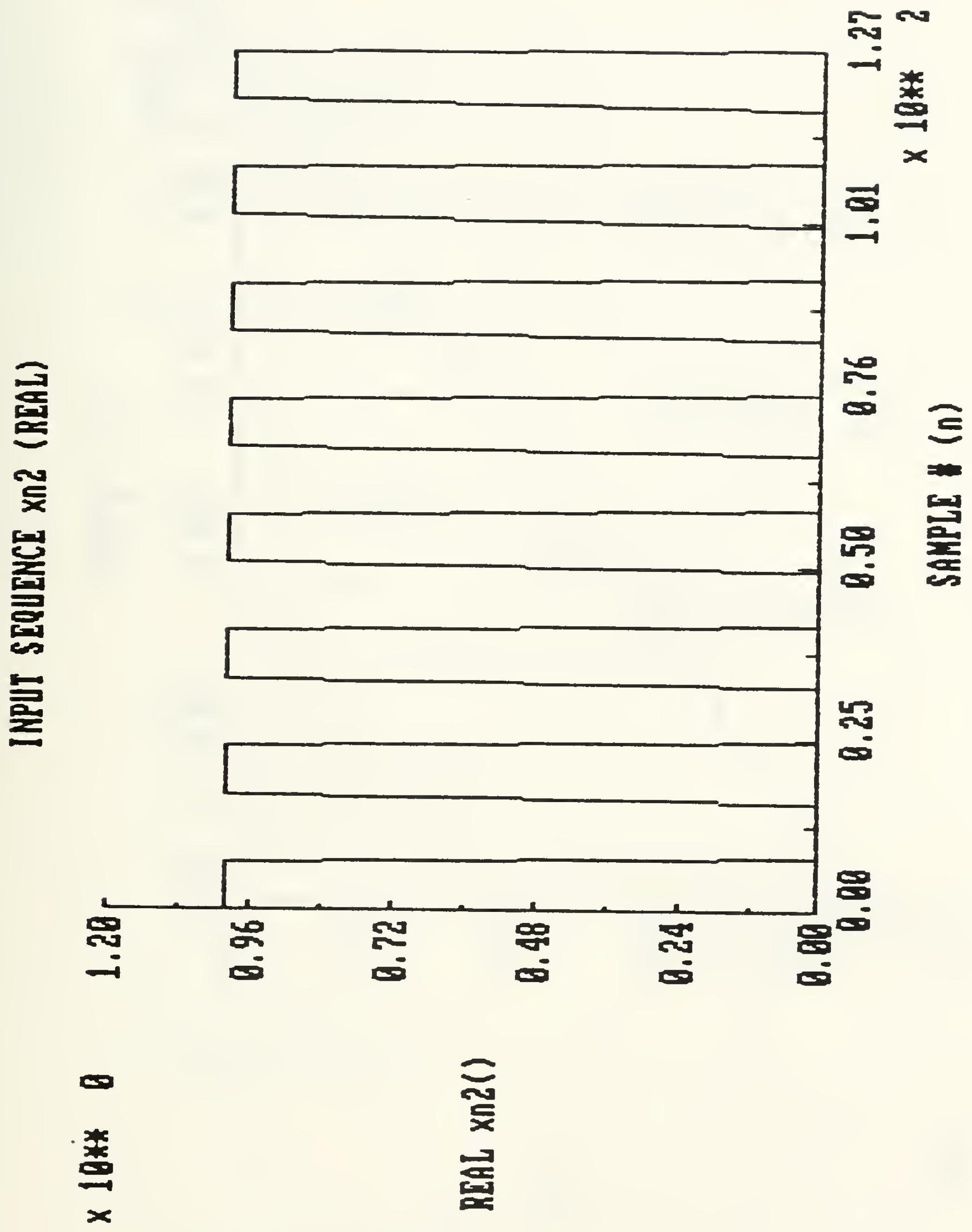


Figure H.5 Input sequence xn2(n) - Example #2.

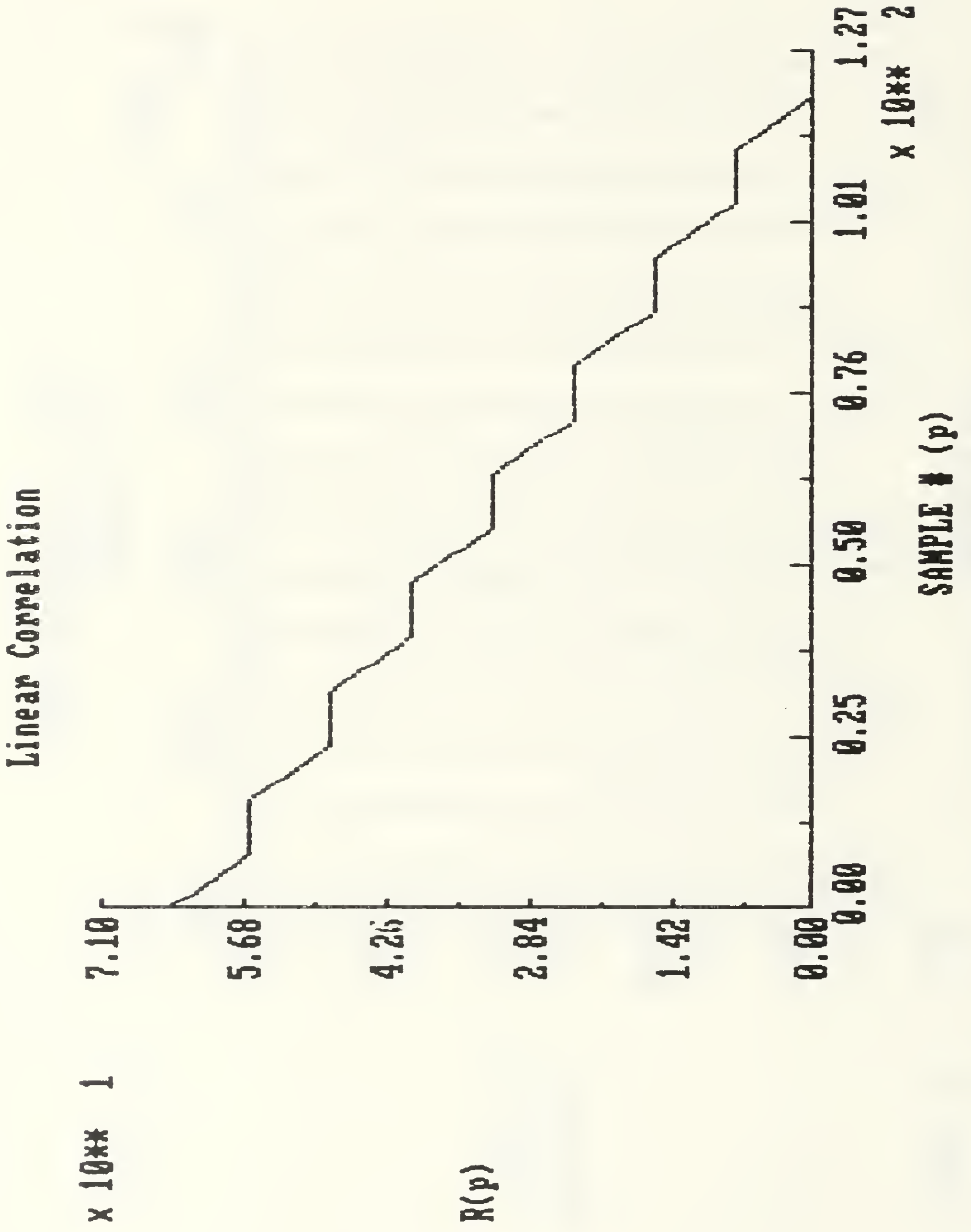


Figure H.6 The result of linear correlation - Example #2.

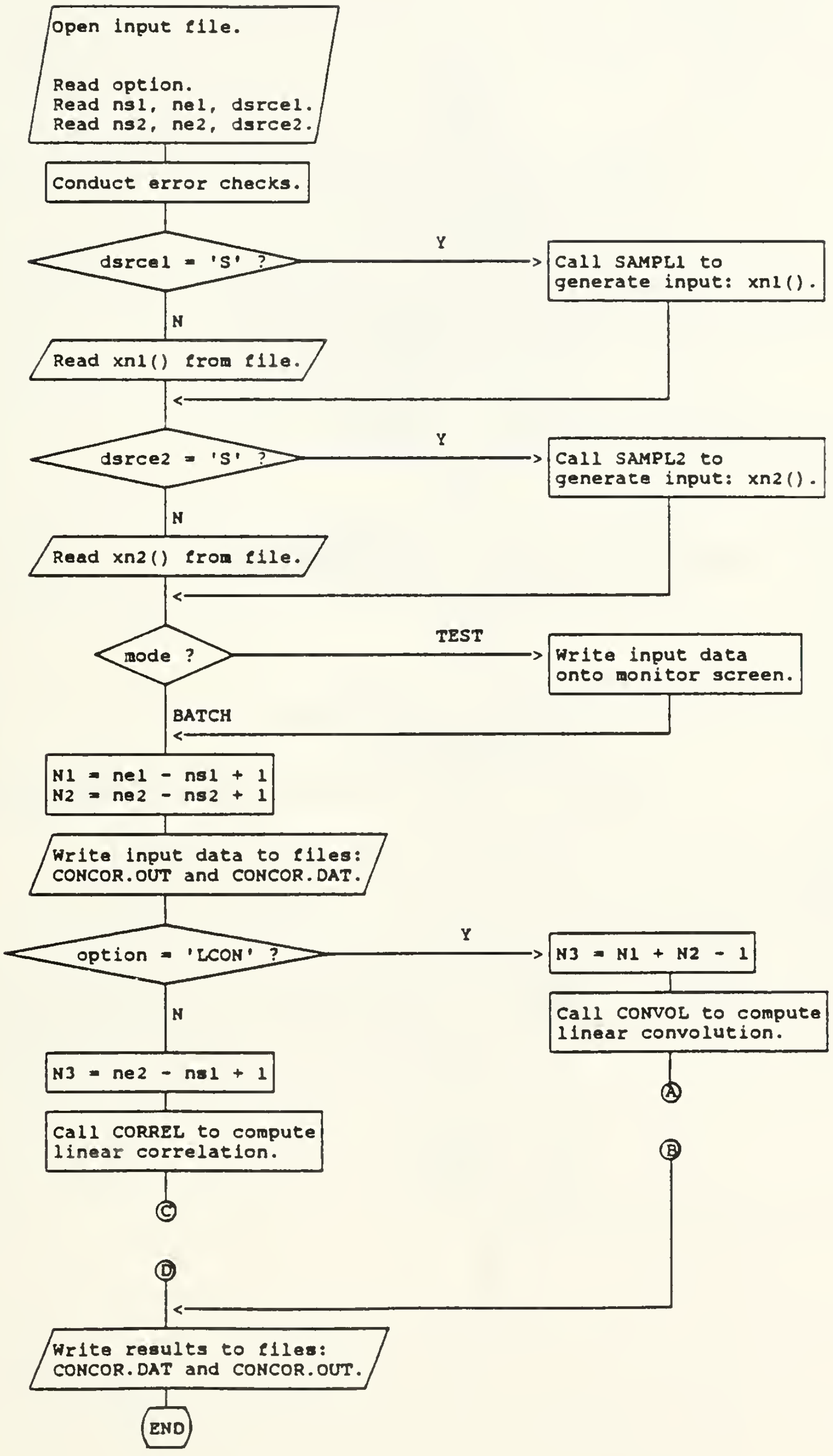

Open input file.
Read option.
Read ns1, ne1, dsrce1.
Read ns2, ne2, dsrce2.
Conduct error checks.
dsrce1 = 'S' ?
Y
N
Call SAMPL1 to generate input: xn1().
Read xn1() from file.
dsrce2 = 'S' ?
Y
N
Call SAMPL2 to generate input: xn2().
Read xn2() from file.
mode ?
TEST
BATCH
Write input data onto monitor screen.
N1 = ne1 - ns1 + 1
N2 = ne2 - ns2 + 1
Write input data to files: CONCOR.OUT and CONCOR.DAT.
option = 'LCON' ?
Y
N
N3 = N1 + N2 - 1
Call CONVOL to compute linear convolution.
N3 = ne2 - ns1 + 1
Call CORREL to compute linear correlation.
A
B
C
D
Write results to files: CONCOR.DAT and CONCOR.OUT.
END

Figure H.7 CONCOR.FOR Software Flowchart.

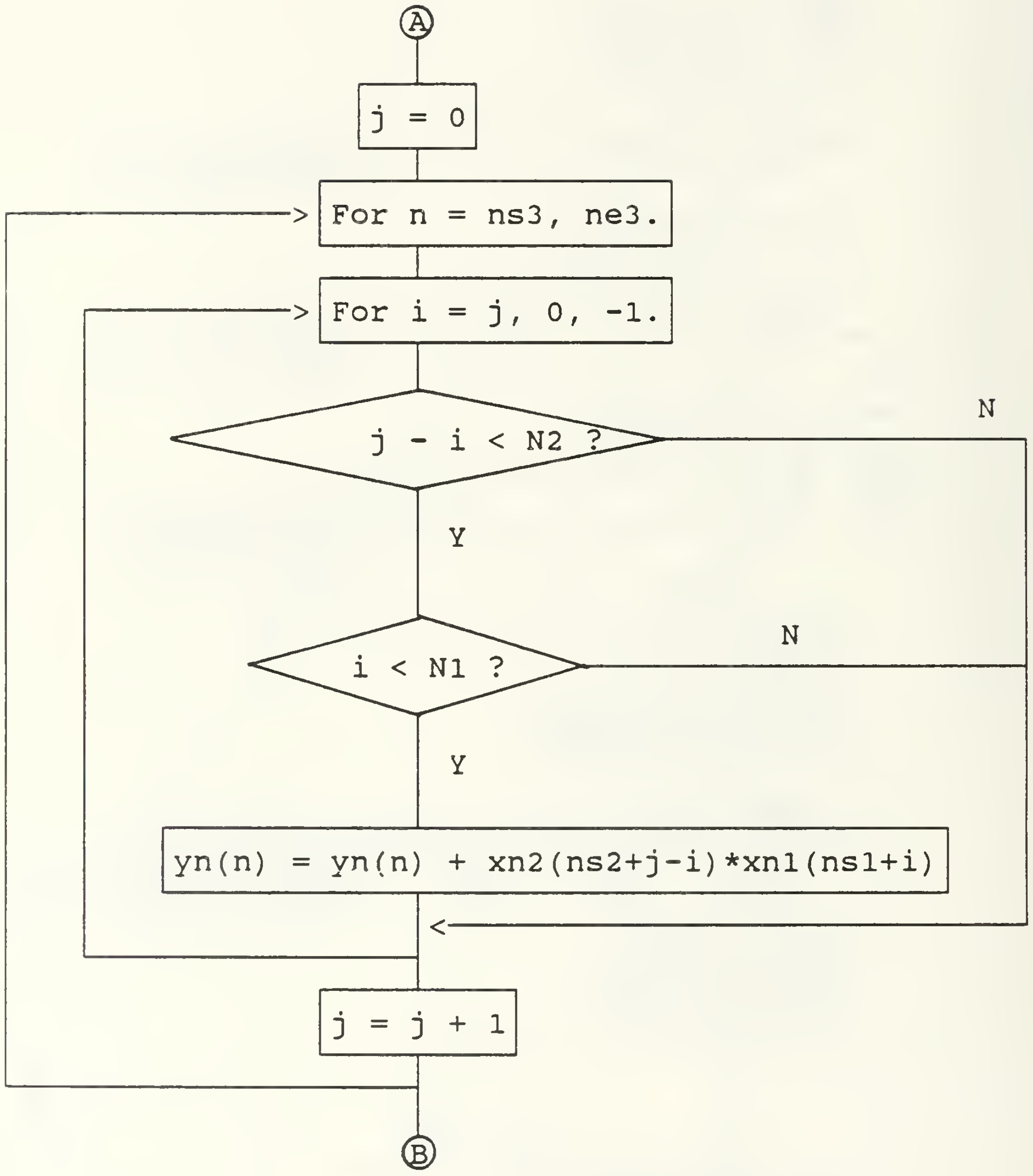


Figure H.8 CONVOL Subroutine Flowchart.

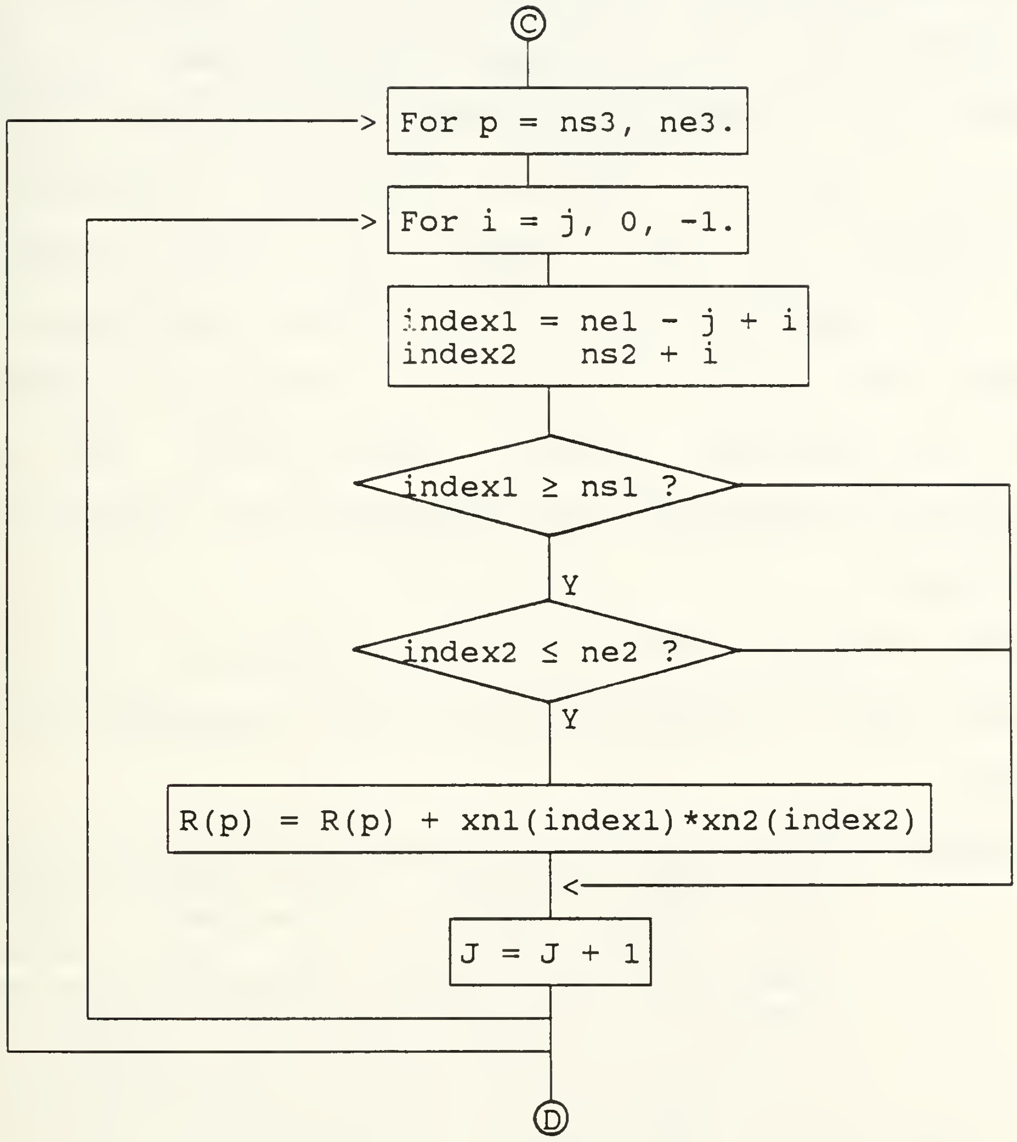


Figure H.9 CORREL Subroutine Flowchart.

Appendix I

The iterative solution to an LTI difference equation is rather straight-forward. If the equation consists of only a few terms, several iterations can be computed by hand calculations. A more complex equation requires a solution derived by an analytical approach or through use of a recursive algorithm. DIFFEQ.FOR provides a recursive means of computing the output sequence y(ns) when provided with the input sequence x(ns) and the initial conditions of the system. This appendix includes the flowcharts of DIFFEQ.FOR and the subroutine DIFFEQ. Additionally, two example problems are developed which demonstrate the capabilities of DIFFEQ.FOR.

The variable names listed below are used in the Fortran source code of DIFFEQ.FOR and in the corresponding flowcharts.

numsys -	The integer value that specifies the number of difference equations in the form of Equation (3.9) that are to be solved. For each difference equation, the input parameters described below must be provided by the user.
L -	The integer value denoting the maximum number of delays occurring in the input sequence x().
N -	The integer value denoting the maximum number of delays occurring in the output sequence y().
nstop -	The integer value corresponding to the largest time index for which the sequence y() is to be solved.
xsorce -	The character string 'F' or 'S' that specifies whether the input sequence x() is to be read from the input file (F) or generated (S) through use of the subroutine XGEN.
b() -	The coefficients of the input sequence corresponding to Equation (3.9).

a() - The coefficients of the output sequence corresponding to Equation (3.9).

y() - The output sequence of length: N + nstop + 1. The initial condition sequence y(-N)...y(-1) must be provided by the user if N > 0. The remaining values in the sequence y(0)...y(nstop) are computed by the program.

x() - The input sequence of length: nstop + 1.

ns - The time index of both the input sequence and the output sequence.

nprob - The integer value corresponding to the difference equation being solved.

Example #1

The first example involves the solution of the difference equation:

$$y(ns) = 1.2*y(ns-1) + 1.5*x(ns)$$

$$\text{Given: } y(-1) = 25.0$$

$$x(ns) = 100.0 \quad \text{for} \quad 0 \leq ns \leq nstop \tag{I.1}$$

The goal is to compute the solution to this difference equation for values of ns in the range: $0 \leq ns \leq 10$. Listed below is the input file DIFFEQ.TST required to run this problem:

DIFFEQ.TST

```
1
000         001         010         F
1.5
1.2
25.0
100.0       100.0       100.0       100.0       100.0       100.0
100.0       100.0       100.0       100.0
```

Included on the page that follows is a listing of the computed solution as it appears in the file DIFFEQ.OUT. The manual computation of y(ns) for the first few values of ns yields the sequence y(ns) = [25 180 366 589.2 ...]. As can be seen from the tabular output, the solution was correctly computed for these values of ns. Continuing with a more analytical approach, the solution to this difference equation can be found for any value of ns ≥ 0 with the aid of the geometric sum equation. The solution, after some manipulation, is:

$$y(ns) = 25.0*1.2^{ns+1} + 150.0* \frac{1.0 - 1.2^{ns+1}}{-0.2} \qquad \text{for } ns \geq 0 \tag{I.2}$$

For example:

$$y(10) = 25.0*1.2^{11} + 150.0* \frac{1.0 - 1.2^{11}}{-0.2} = 5008.315 \tag{I.3}$$

To an accuracy of two decimal places the computed solution matches the analytical solution.

This example problem is also developed in the header text of DIFFEQ.FOR and can be run by the user in Test Mode by using the prestored data found in the input file DIFFEQ.TST.

DIFFEQ.OUT

```
              INPUT DATA FOR PROBLEM #  1

 PROBLEM # 1    INPUT DATA SOURCEFILE: DIFFEQ.TST
 THE NUMBER OF INPUT DELAYS: L =   0
 THE NUMBER OF OUTPUT DELAYS: N =   1
 THE VALUE OF nstop IS:  10
 THE COEFFICIENTS b(0), b(1), ..., b(L) ARE:

.150000E+01


 THE COEFFICIENTS a(1), ..., a(N) ARE:

.120000E+01


            OUTPUT DATA FOR PROBLEM #  1

   ns         x(ns)              y(ns)
   -1      .000000E+00        .250000E+02
    0      .100000E+03        .180000E+03
    1      .100000E+03        .366000E+03
    2      .100000E+03        .589200E+03
    3      .100000E+03        .857040E+03
    4      .100000E+03        .117845E+04
    5      .100000E+03        .156414E+04
    6      .100000E+03        .202697E+04
    7      .100000E+03        .258236E+04
    8      .100000E+03        .324883E+04
    9      .100000E+03        .404860E+04
   10      .100000E+03        .500832E+04

---------------- END OF PROBLEM # 1  ----------------
```

Example #2

This second example requires the solution to the difference equation:

$$y(ns) = 0.95*y(ns-1) - 0.9025*y(ns-2) + x(ns) - .475*x(ns-1) \tag{I.4}$$

Given: $y(-2) = y(-1) = 0.0$

$$x(ns) = \begin{cases} 1.0 & \text{for } ns = 0 \\ 0.0 & \text{otherwise} \end{cases}$$

The system described by this difference equation corresponds to the transfer function:

$$H(z) = \frac{y(z)}{x(z)} = \frac{1.0 - .475*z^{-1}}{1.0 - 0.95z^{-1} + 0.9025z^{-2}} \tag{I.5}$$

With the aid of the Inverse z-Transform Formula, the analytical solution of this example problem is found to be:

$$y(ns) = 0.95^{ns} * \cos(\pi*ns/3.0) \qquad \text{for } ns \geq 0 \tag{I.6}$$

The next page of this appendix is a plot of the output sequence for values of ns in the range: $0 \leq ns \leq 80$. The plot clearly shows both the decaying envelope of the sequence, as well as the constant frequency sinusoid.

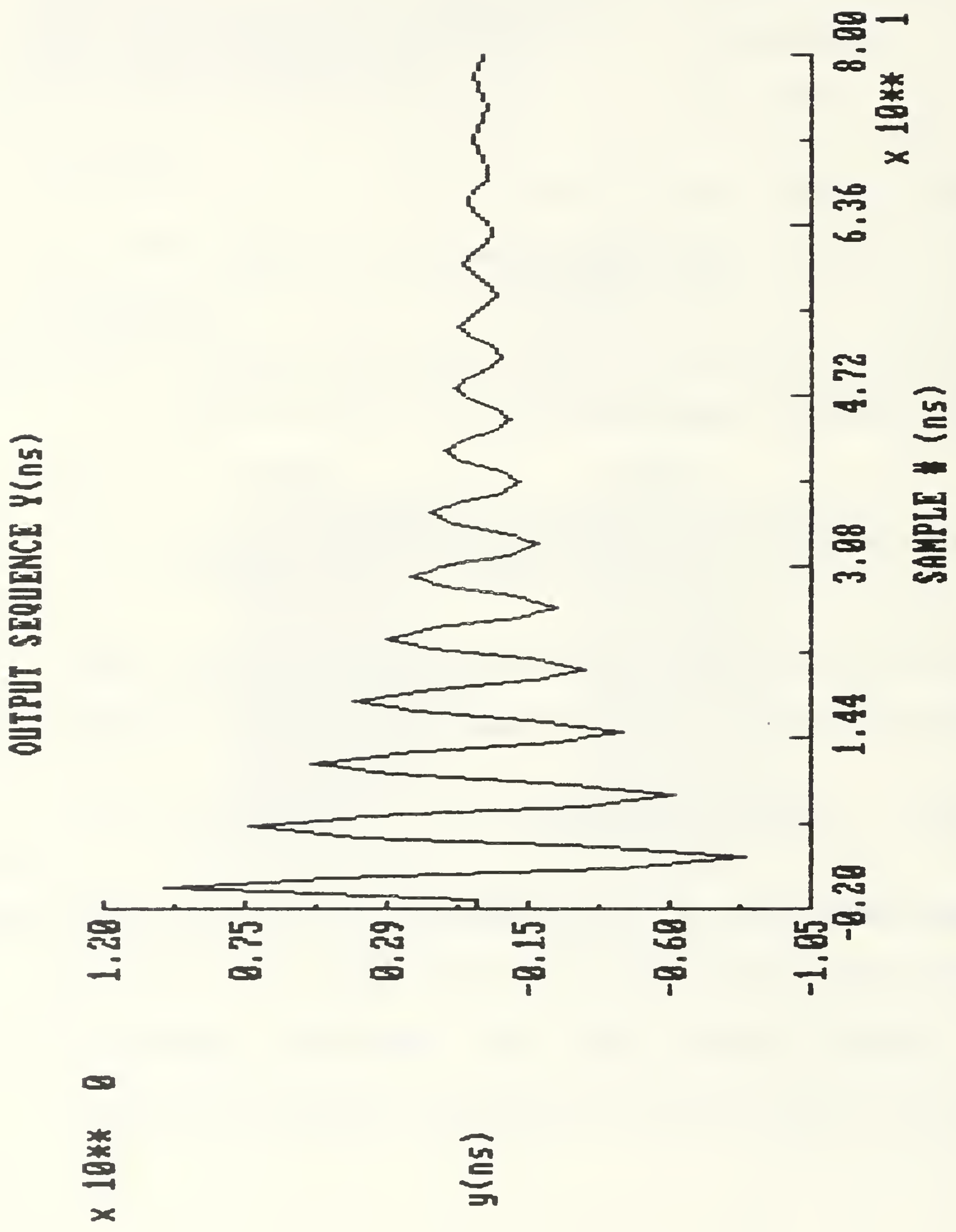
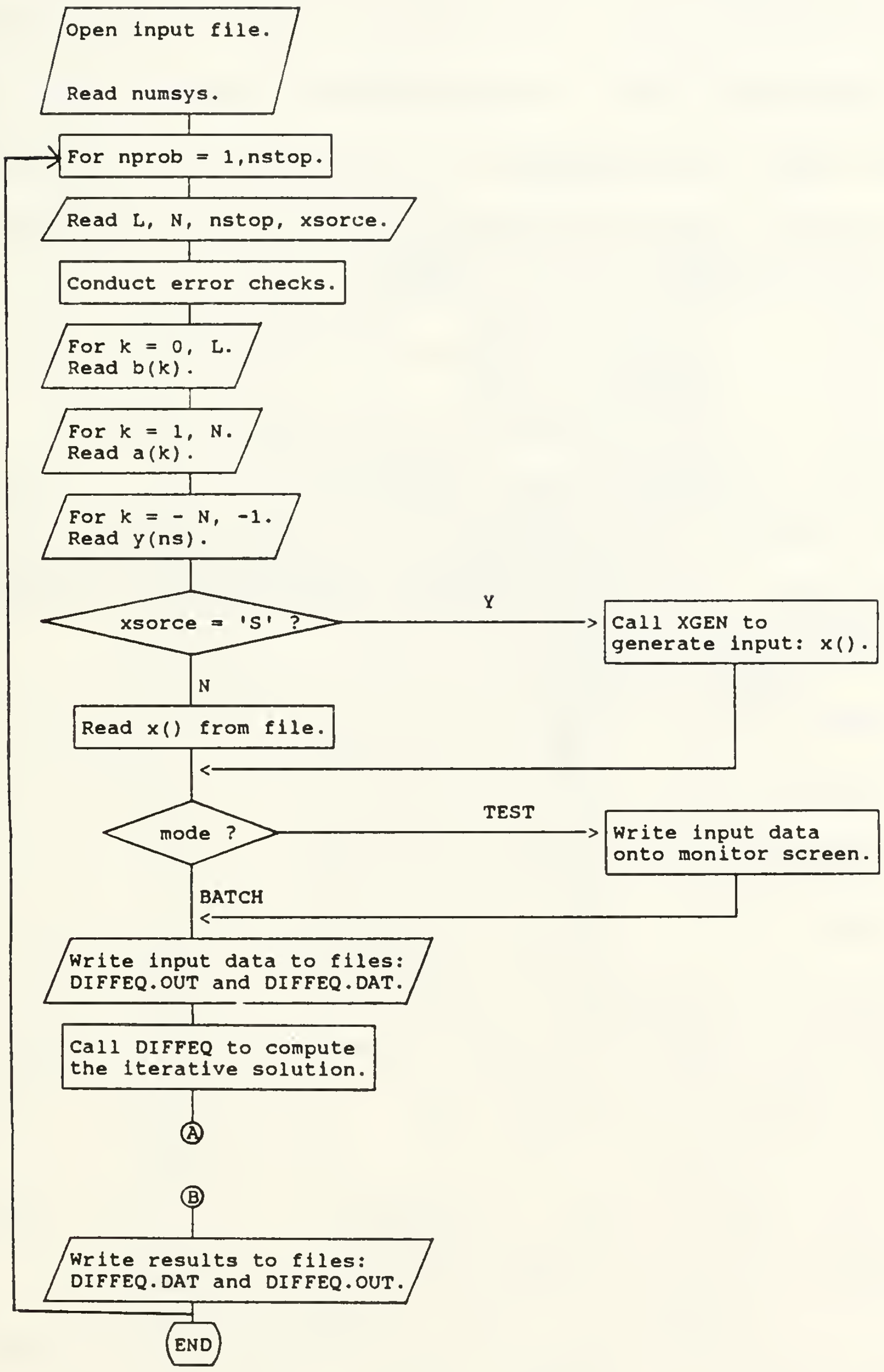


Figure I.1 System output - Example #2.



Open input file.
Read numsys.
For nprob = 1,nstop.
Read L, N, nstop, xsorce.
Conduct error checks.
For k = 0, L.
Read b(k).
For k = 1, N.
Read a(k).
For k = - N, -1.
Read y(ns).
xsorce = 'S' ?
Y
Call XGEN to generate input: x().
N
Read x() from file.
mode ?
TEST
Write input data onto monitor screen.
BATCH
Write input data to files: DIFFEQ.OUT and DIFFEQ.DAT.
Call DIFFEQ to compute the iterative solution.
A
B
Write results to files: DIFFEQ.DAT and DIFFEQ.OUT.
END

Figure I.2 DIFFEQ.FOR Software Flowchart.

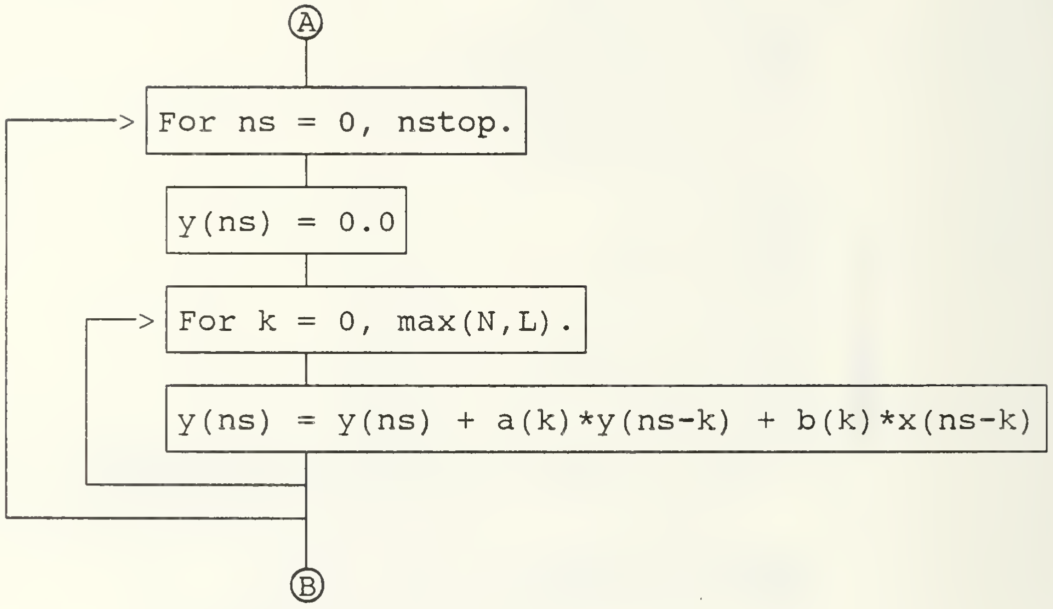


Figure I.3 DIFFEQ Subroutine Flowchart.

Appendix J

Included in this appendix are two example problems that demonstrate the capabilities of the program STATEQ.FOR. The final pages of this appendix are the software flowcharts of the main program STATEQ.FOR and the subroutine ITRATE. Listed below are the names of the variables used throughout the flowcharts and the corresponding Fortran source code.

N -	An integer value that specifies the number of system states.
M -	An integer value that specifies the number of system inputs.
Q -	An integer value that specifies the number of system outputs.
nstop -	The integer value corresponding to the largest time index for which the system of state equations is to be solved.
xsorce -	The character string 'F' or 'S' that specifies whether the input sequence xs() is to be read from the input file (F) or generated (S) through use of the subroutine XGEN.
A -	An N x N matrix of state coefficients as they occur in Equation (3.10).
B -	An N x M matrix of input coefficients as they occur in Equation (3.10).
C -	A Q x N matrix of output coefficients as they occur in Equation (3.11).
D -	A Q x M matrix of input coefficients as they occur in Equation (3.11).
v() -	An N x 1 vector consisting of values that describe the initial condition of the system.
ns -	An integer value denoting the time index.
xs(i,ns) -	An M x (nstop+1) array consisting of the input sequence(s). The index i denotes the input number (1, ..., M), and the index ns denotes the sample number (0, 1, ..., nstop).
vs(i,ns) -	An N x (nstop+1) array consisting of the state(s) of the system. The index i denotes the state (1, ..., Q), and the index ns denotes the sample number (0, 1, ..., nstop).
ys(i,ns) -	A Q x (nstop+1) array consisting of the output sequence(s). The index i denotes the output number (1, ..., Q), and the index ns denotes the sample number (0, 1, ..., nstop).

xi - A dummy variable that stores the weighted cumulative contribution of the input sequence(s) for each value of ns.

Example #1

This first example problem demonstrates the iterative solution to the state equations:

$$v(ns+1) = \begin{bmatrix} 0.0 & -1.0 \\ 1.0 & 0.0 \end{bmatrix} v(ns) + \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix} x(ns) \tag{J.1}$$

$$y(ns) = [\ 1.0 \quad -1.0\ ]\ v(ns) + [\ 1.0\ ]\ x(ns) \tag{J.2}$$

The initial condition vector is:

$$v(0) = \begin{bmatrix} 5.0 \\ -5.0 \end{bmatrix} \tag{J.3}$$

The input vector is:

$$x(ns) = [\ 10.0\ ] \qquad \text{for } 0 \le ns \le 3 \tag{J.4}$$

The goal of this problem is to compute the solution to the given system of equations for values of ns in the range: $0 \le ns \le 3$. This example problem also appears in the header text of STATEQ.FOR and can be run by the user in Test Mode by using the input data prestored in the file STATEQ.TST. A listing of STATEQ.TST appears below.

STATEQ.TST

```
02          1          1
03          F
0.0         -1.0
1.0         0.0
1.0
0.0
1.0         -1.0
1.0
5.0         -5.0
10.0
10.0
10.0
10.0
```

Manual calculation of the solution to this problem yields the following sequences:

$$v1(ns) = [\ 5.0 \quad 15.0 \quad 5.0 \quad -5.0\ ] \tag{J.5}$$

$$v2(ns) = [\ -5.0 \quad 5.0 \quad 15.0 \quad 5.0\ ] \tag{J.6}$$

$$y1(ns) = [\ 20.0 \quad 20.0 \quad 0.0 \quad 0.0\ ] \tag{J.7}$$

A listing of the tabular output file STATEQ.OUT follows. As the tabular output indicates, the sequences were correctly computed over the specified range of ns.

STATEQ.OUT

```
           INPUT PARAMETERS:

INPUT DATA SOURCE FILE: STATEQ.TST
THE NUMBER OF STATES IS: N =  2
THE NUMBER OF SYSTEM INPUTS IS: M = 1
THE NUMBER OF SYSTEM OUTPUTS IS: Q = 1
THE VALUE OF nstop IS: nstop =   3
THE VALUE FOR xsorce IS: F


THE MATRIX A(i,j) IS:

.0000E+00  -.1000E+01
.1000E+01   .0000E+00


THE MATRIX B(i,j) IS:

.1000E+01
.0000E+00


THE MATRIX C(i,j) IS:

.1000E+01  -.1000E+01


THE MATRIX D(i,j) IS:

.1000E+01


THE INITIAL CONDITION OF THE STATE VECTOR IS:

  v1 =  .500000E+01
  v2 = -.500000E+01
```

OUTPUT DATA:

```
FOR ns =  0 THE STATE OF THE SYSTEM IS:
  THE VECTOR x is:
    x1 =  .100000E+02
  THE VECTOR v is:
    v1 =  .500000E+01
    v2 = -.500000E+01
  THE VECTOR y is:
    y1 =  .200000E+02

FOR ns =  1 THE STATE OF THE SYSTEM IS:
  THE VECTOR x is:
    x1 =  .100000E+02
  THE VECTOR v is:
    v1 =  .150000E+02
    v2 =  .500000E+01
  THE VECTOR y is:
    y1 =  .200000E+02

FOR ns =  2 THE STATE OF THE SYSTEM IS:
  THE VECTOR x is:
    x1 =  .100000E+02
  THE VECTOR v is:
    v1 =  .500000E+01
    v2 =  .150000E+02
  THE VECTOR y is:
    y1 =  .000000E+00

FOR ns =  3 THE STATE OF THE SYSTEM IS:
  THE VECTOR x is:
    x1 =  .100000E+02
  THE VECTOR v is:
    v1 = -.500000E+01
    v2 =  .500000E+01
  THE VECTOR y is:
    y1 =  .000000E+00
```

Example #2

A low-pass filter having a zero at z = -1.0 and poles at $z = .95,\ .95e^{-j\pi/6},\ .95e^{+j\pi/6}$ has the transfer function:

$$H(z) = \frac{z + 1}{z^3 - 2.5954z^2 + 2.4657z - .8574} \tag{J.8}$$

A state-matrix representation of this system is:

$$\mathbf{v}(ns+1) = \begin{bmatrix} 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \\ 0.8574 & -2.4657 & 2.5954 \end{bmatrix} \mathbf{v}(ns) + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \mathbf{x}(ns) \tag{J.9}$$

$$\mathbf{y}(ns) = [\ 1.0 \quad 1.0 \quad 0.0\ ]\ \mathbf{v}(ns) + [\ 0\ 0\ 0\ ]\ \mathbf{x}(ns) \tag{J.10}$$

The inputs to the system are the three sequences:

$$x1(ns) = 10.0*u(ns) \tag{J.11}$$

$$x2(ns) = 2*\cos(\pi*ns/6)*u(ns) \tag{J.12}$$

$$x3(ns) = 2*\cos(\pi*ns/2)*u(ns) \tag{J.13}$$

The initial condition of the system is the vector:

$$\mathbf{v}() = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{J.14}$$

To solve this problem, the input file STATEQ.IN was created. A listing of this file appears below. Plots of the sequences x1(ns), x2(ns), x3(ns), v1(ns), v2(ns), v3(ns), and y1(ns) are included on the pages that follow. The sequences were computed for 100 values of ns (nstop = 99). Because of the lengths of the input sequences, all three input sequences were generated through use of the subroutine XGEN.

As the plots indicate, the filter amplifies the low frequency inputs x1 and x2, but attenuates the high frequency input x3.

<u>STATEQ.IN</u>

```
03          3           1
99          S
0.0         1.0         0.0
0.0         0.0         1.0
0.8574      -2.4657     2.5954
0.0         0.0         0.0
0.0         0.0         0.0
1.0         1.0         1.0
1.0         1.0         0.0
0.0         0.0         0.0
0.0         0.0         0.0
```

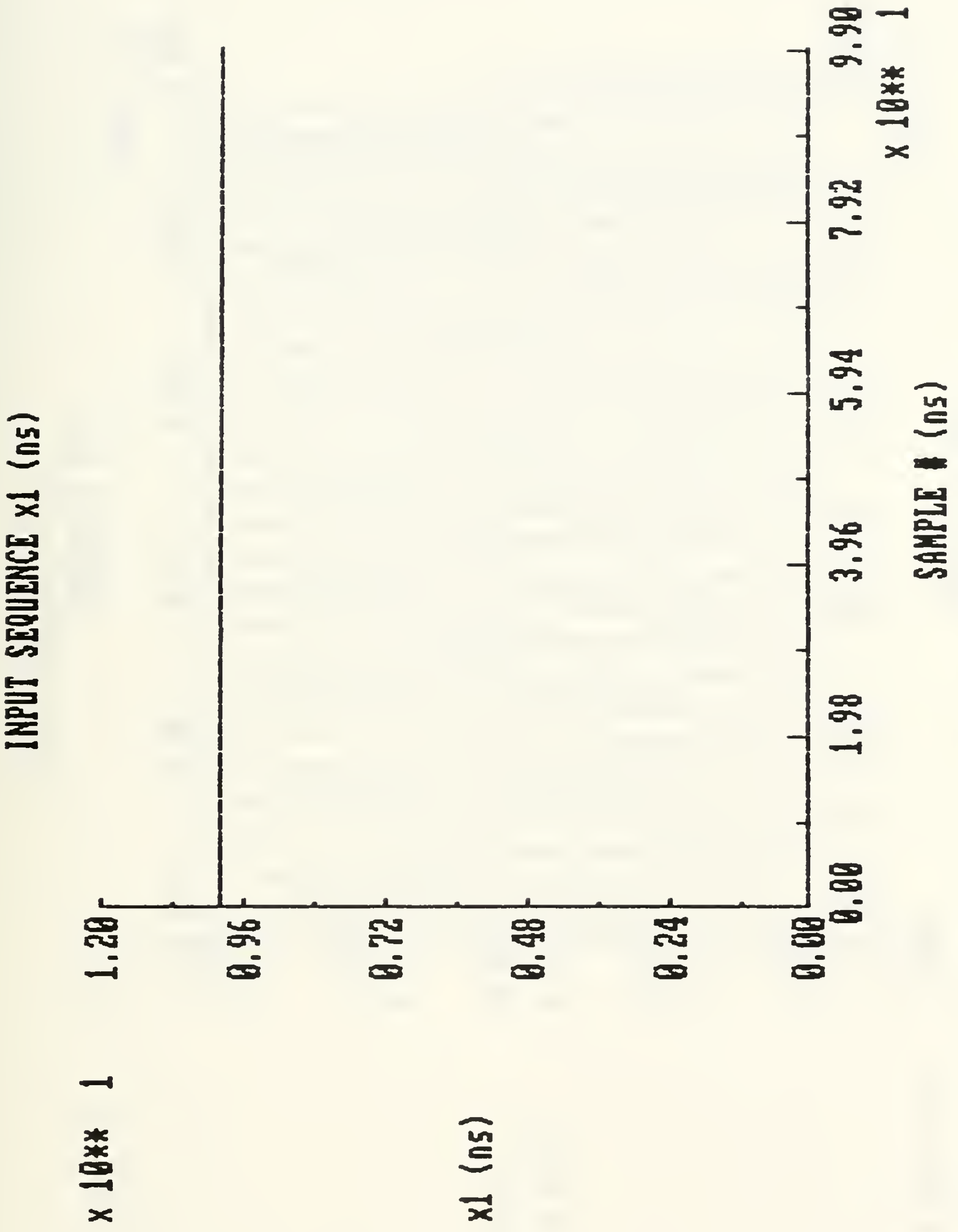


Figure J.1 Input sequence x1(ns) - Example #2.

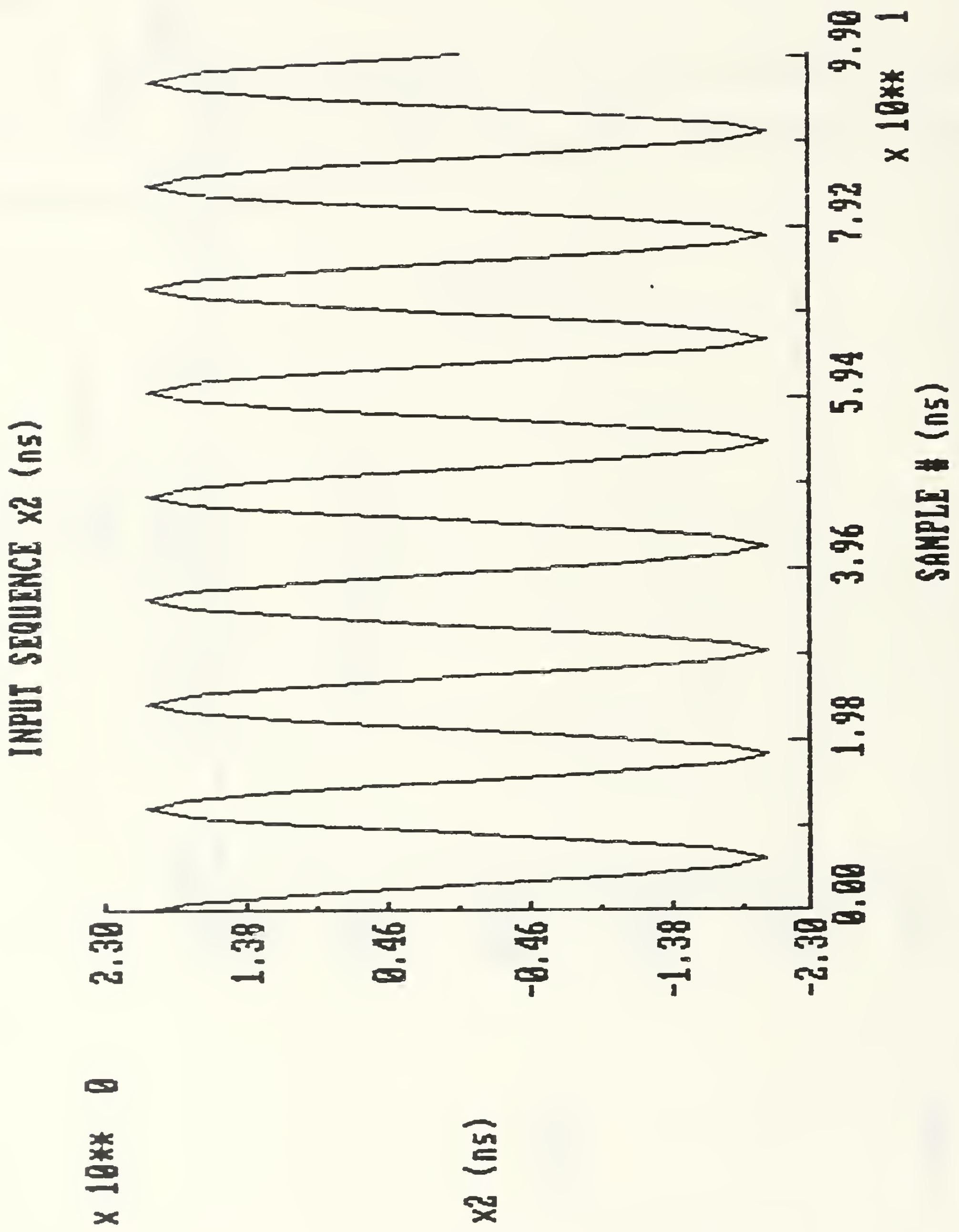


Figure J.2 Input sequence x2(ns) - Example #2.

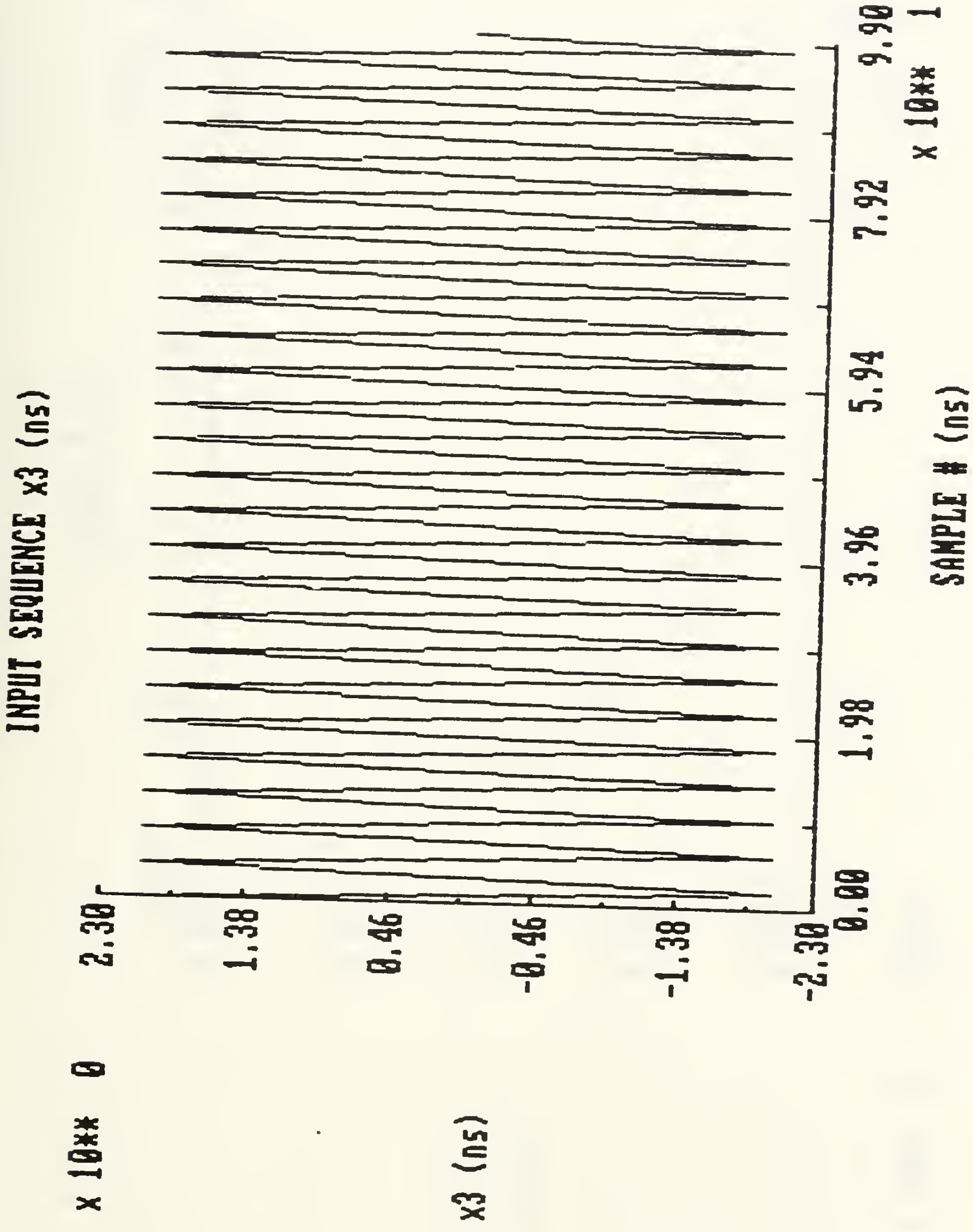


Figure J.3 Input sequence x3(ns) - Example #2.

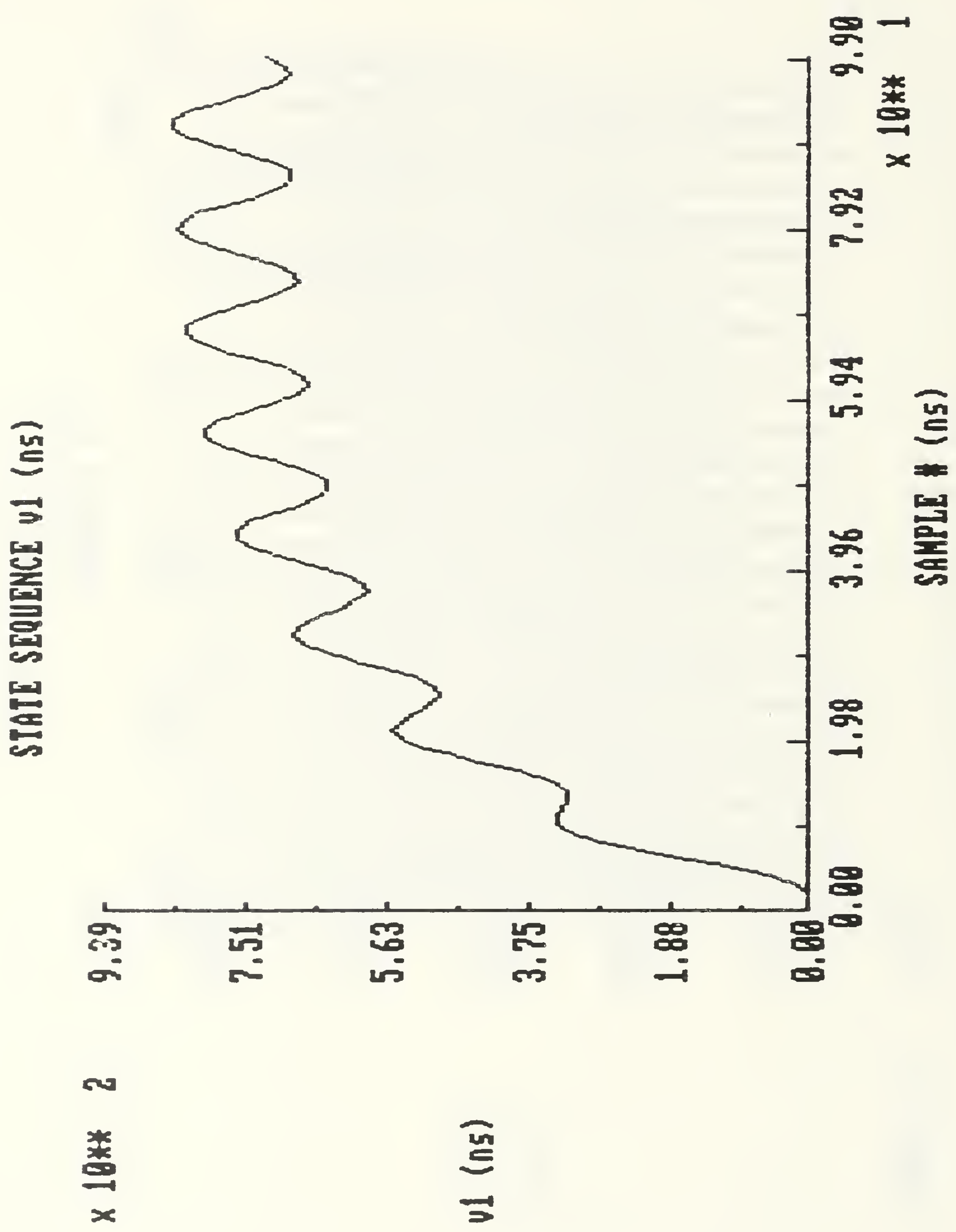


Figure J.4 State sequence v1(ns) - Example #2.

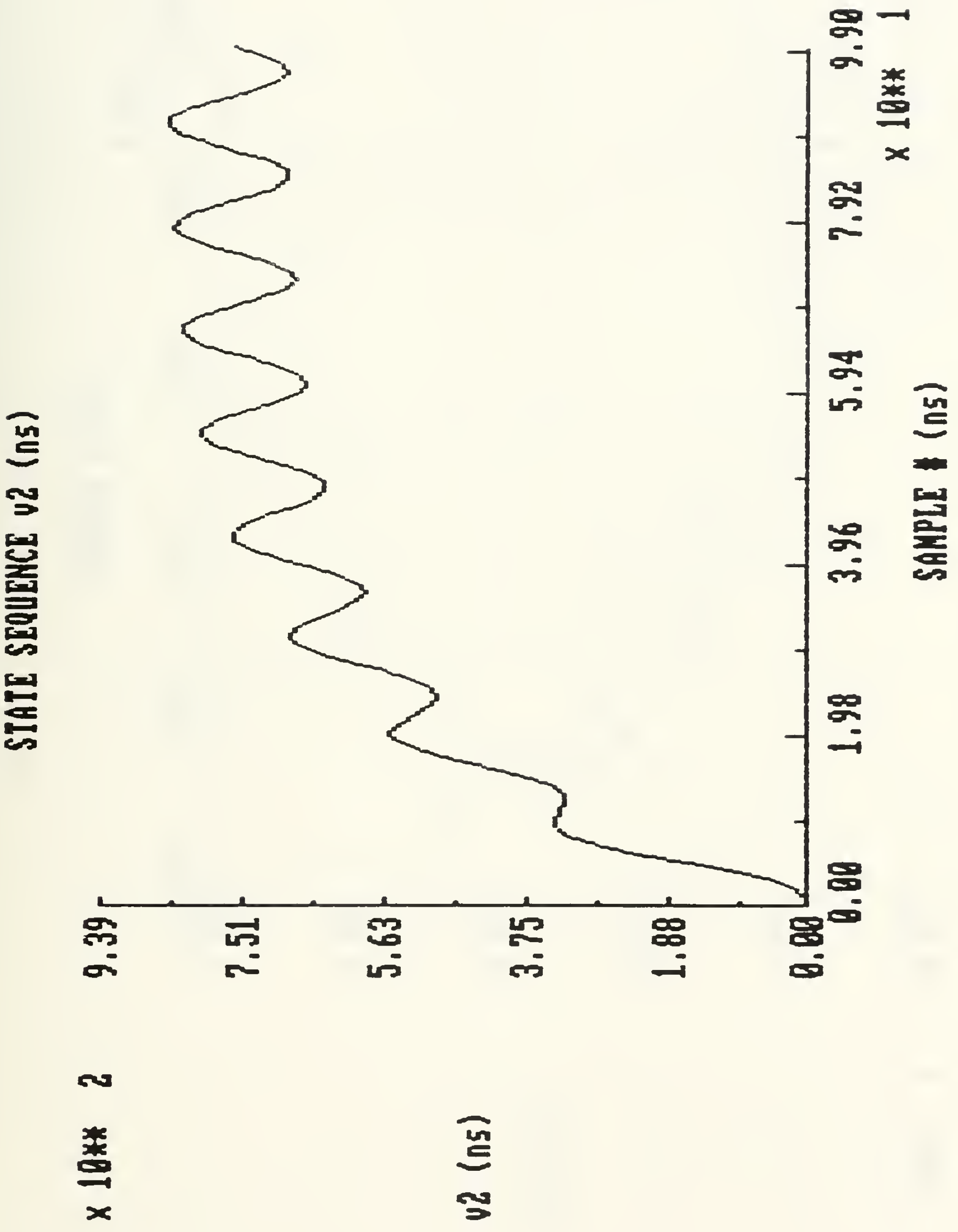


Figure J.5 State sequence v2(ns) - Example #2.

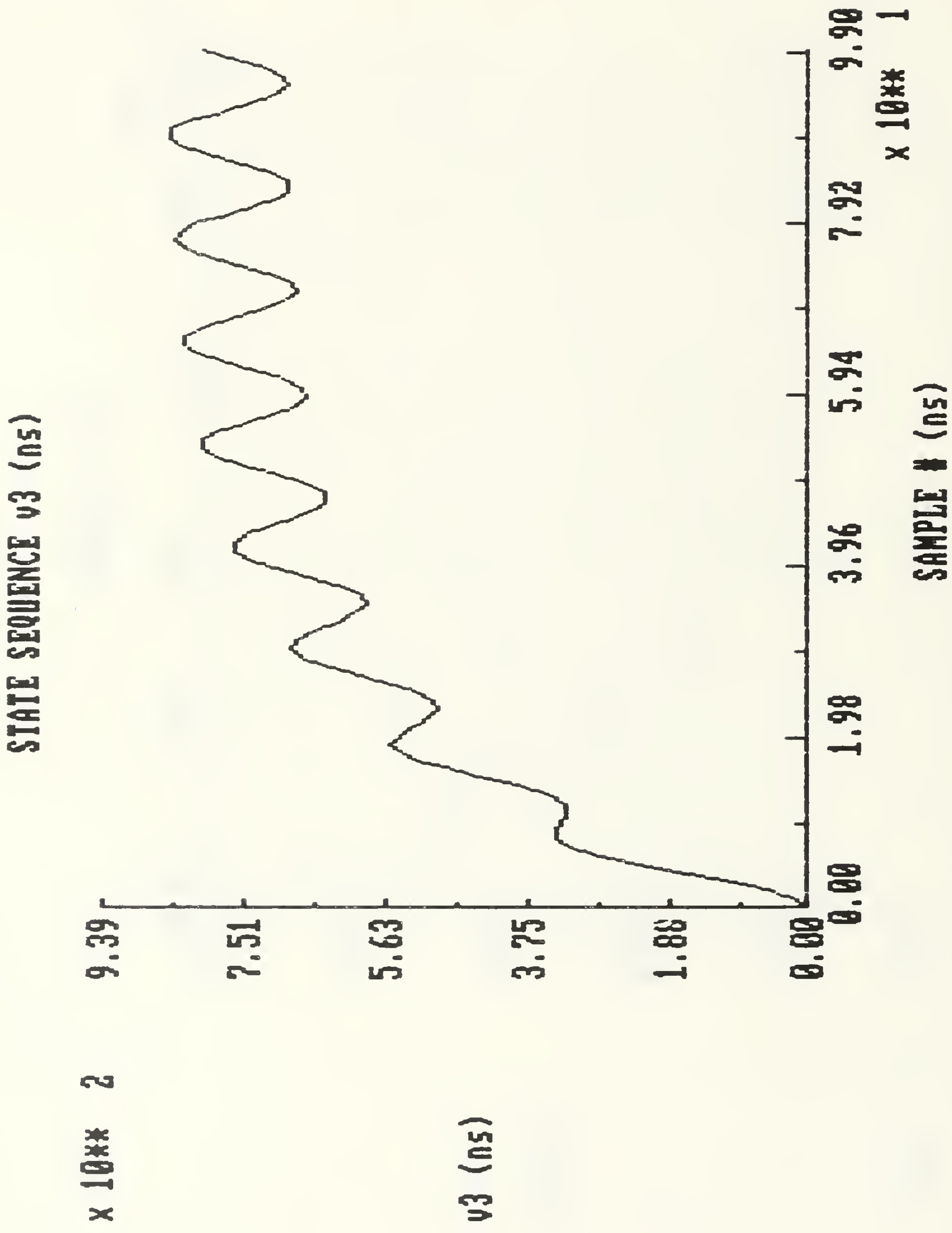


Figure J.6 State sequence v3(ns) - Example #2.

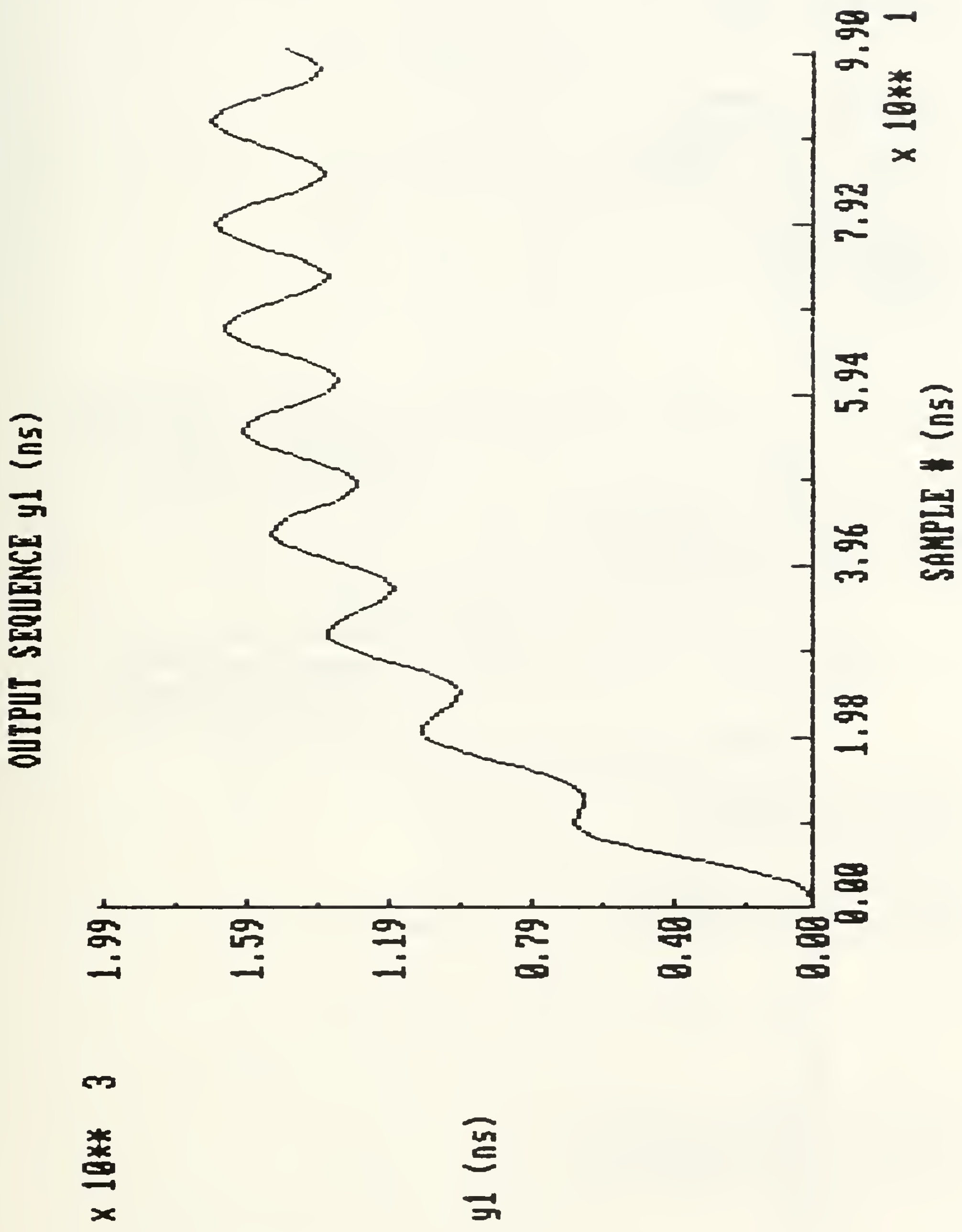
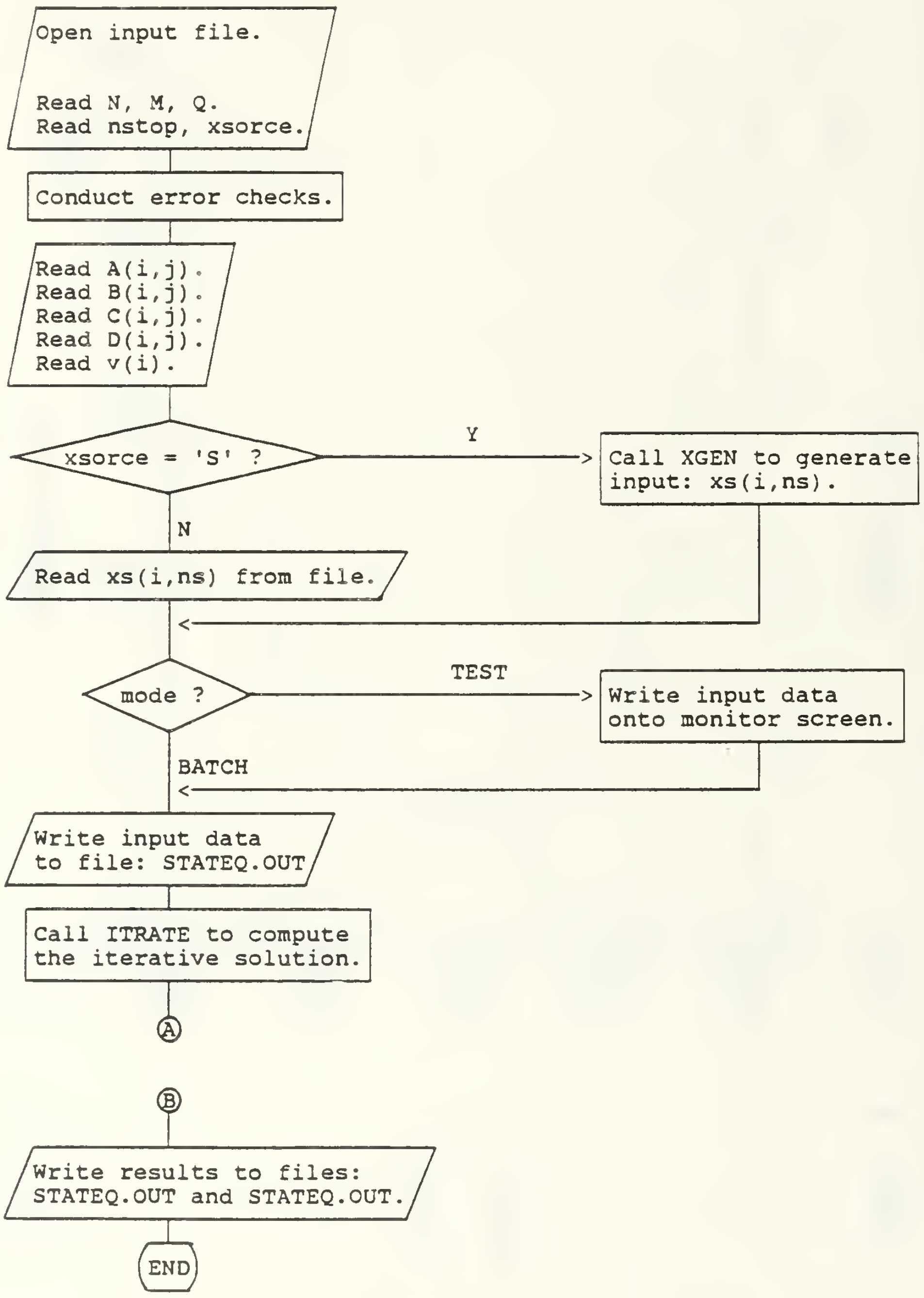


Figure J.7 Output sequence y1(ns) - Example #2.



Open input file.
Read N, M, Q.
Read nstop, xsorce.
Conduct error checks.
Read A(i,j).
Read B(i,j).
Read C(i,j).
Read D(i,j).
Read v(i).
xsorce = 'S' ?
Y
Call XGEN to generate input: xs(i,ns).
N
Read xs(i,ns) from file.
mode ?
TEST
Write input data onto monitor screen.
BATCH
Write input data to file: STATEQ.OUT
Call ITRATE to compute the iterative solution.
A
B
Write results to files: STATEQ.OUT and STATEQ.OUT.
END

Figure J.8 STATEQ.FOR Software Flowchart.

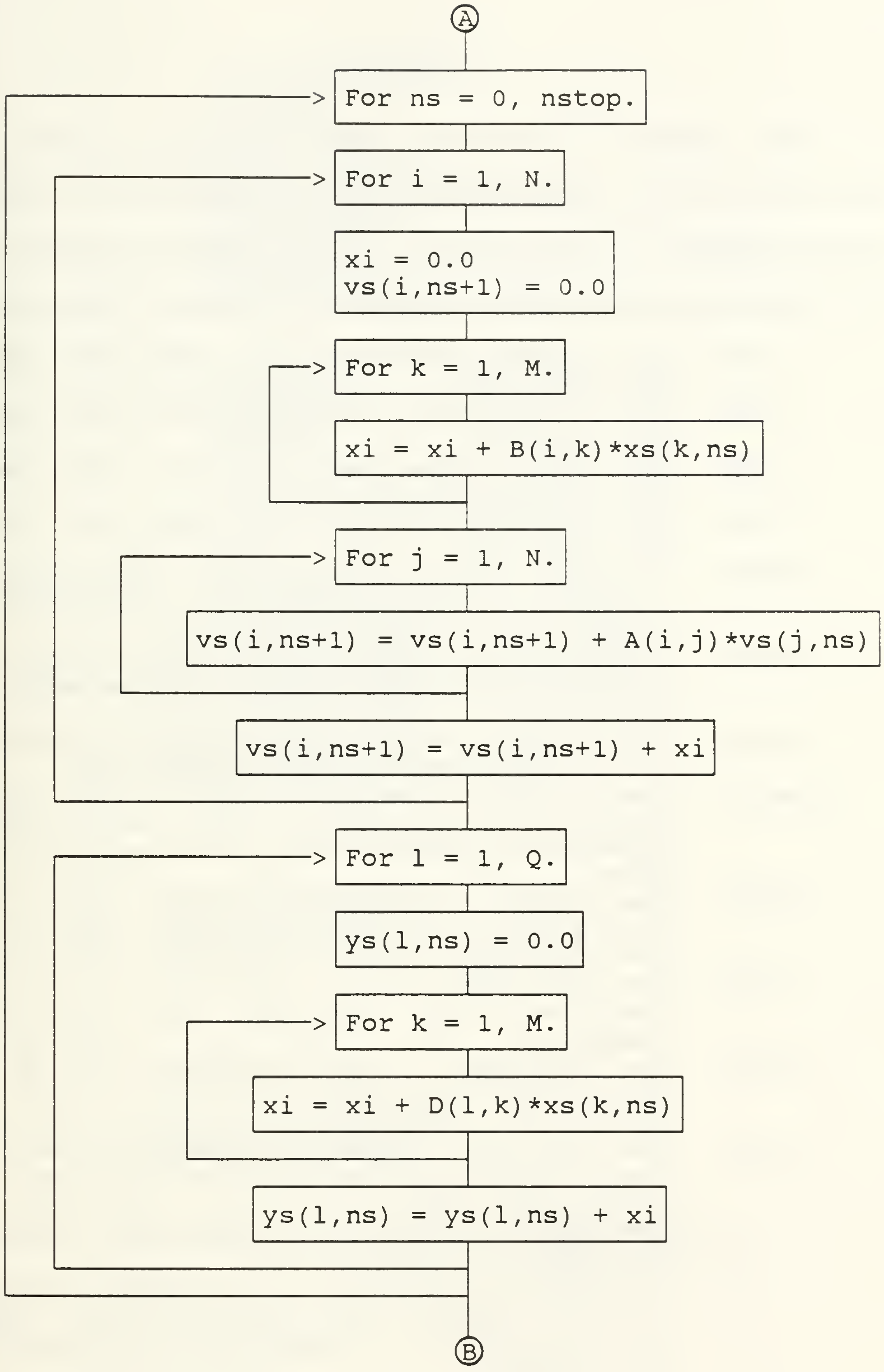
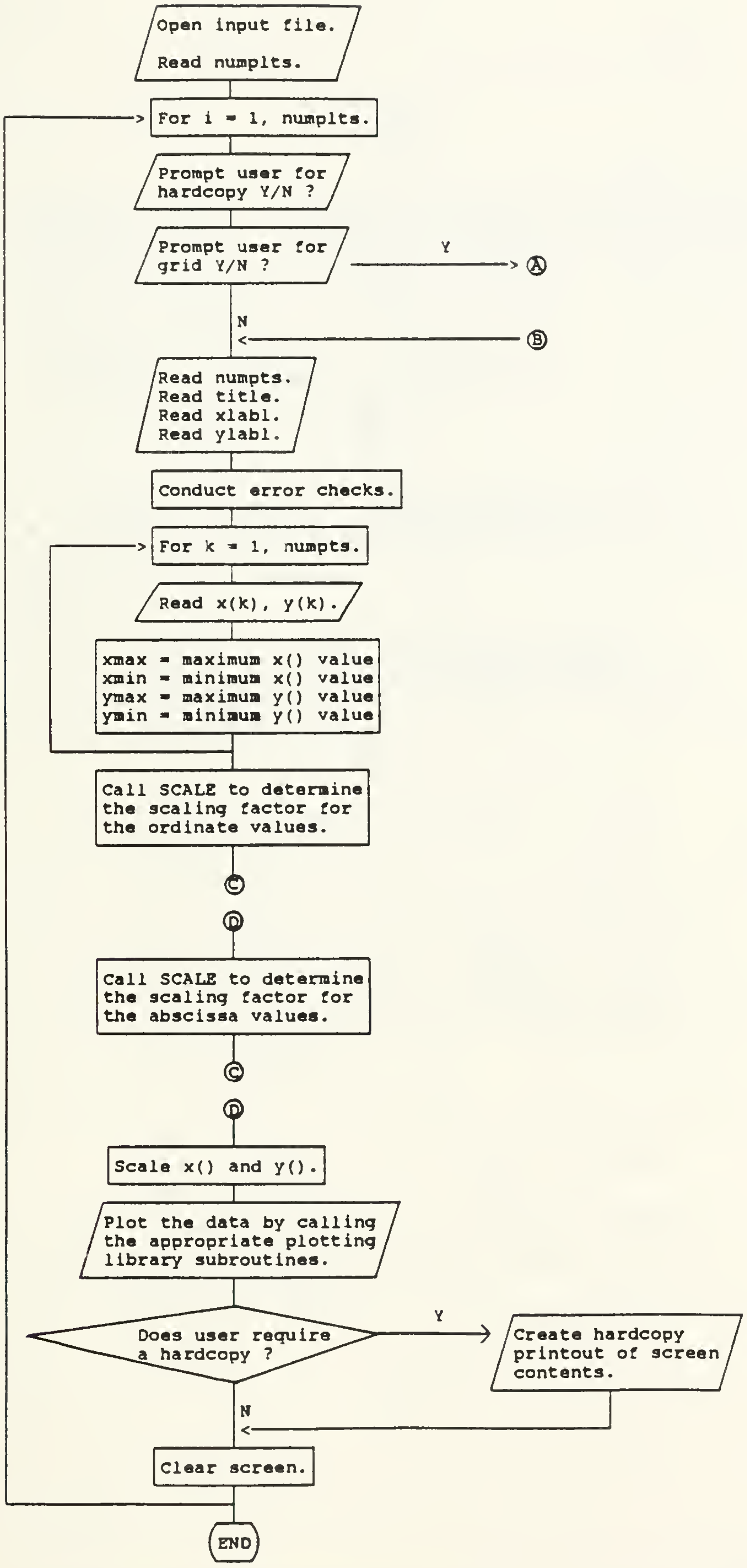


Figure J.9 ITRATE Subroutine Flowchart.

Appendix K

The software flowcharts describing the program PLOT-DAT.FOR and the subroutines SCALE and GRIDD are included in this appendix. Listed below are the variable names used in the flowcharts and the corresponding Fortran source code.

xmax -	The maximum ordinate value read from the data for each plot.
xmin -	The minimum ordinate value read from the data for each plot.
ymax -	The maximum abscissa value read from the data for each plot.
ymin -	The minimum abscissa value read from the data for each plot.
valmax -	A dummy variable passed to subroutine SCALE containing the maximum value of an array to be scaled (i.e., xmax or ymax).
valmin -	A dummy variable passed to subroutine SCALE containing the minimum value of an array to be scaled (i.e., xmin or ymin).
iscal -	An integer value that contains the scaling value determined by subroutine SCALE.
numplts -	The integer value that specifies the number of plots to be created. For each plot, the parameters listed below must occur in the input file.
numpts -	The integer value that specifies the number of data points to be read from the input file for the given plot.
title -	The character string consisting of up to 40 characters that comprise the title of the graph.
xlabl -	The character string consisting of up to 14 characters that comprise the label for the x-axis of the graph.
ylabl -	The character string consisting of up to 14 characters that comprise the label for the y-axis of the graph.
x() -	The array containing the ordinate values read from the input file.
y() -	The array containing the abscissa values read from the input file. Each ordinate, abscissa pair corresponds to one point to be plotted by the program.



Open input file.
Read numplts.
For i = 1, numplts.
Prompt user for hardcopy Y/N ?
Prompt user for grid Y/N ?
Y
A
N
B
Read numpts.
Read title.
Read xlabl.
Read ylabl.
Conduct error checks.
For k = 1, numpts.
Read x(k), y(k).
xmax = maximum x() value
xmin = minimum x() value
ymax = maximum y() value
ymin = minimum y() value
Call SCALE to determine the scaling factor for the ordinate values.
C
D
Call SCALE to determine the scaling factor for the abscissa values.
C
D
Scale x() and y().
Plot the data by calling the appropriate plotting library subroutines.
Does user require a hardcopy ?
Y
Create hardcopy printout of screen contents.
N
Clear screen.
END

Figure K.1 PLOTDAT.FOR Software Flowchart.

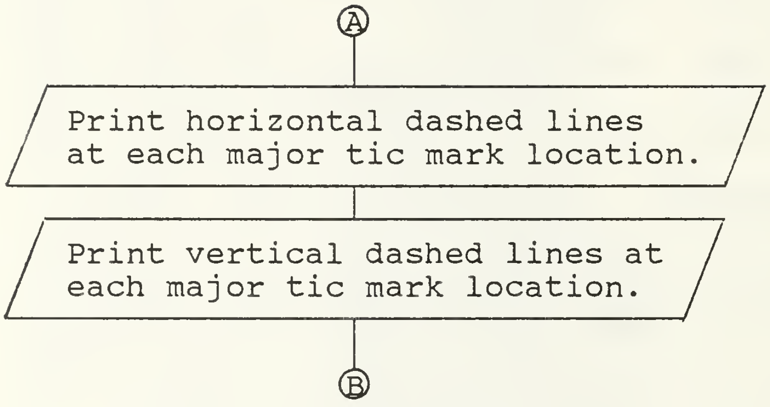

A
Print horizontal dashed lines at each major tic mark location.
Print vertical dashed lines at each major tic mark location.
B

Figure K.2 GRIDD Subroutine Flowchart.

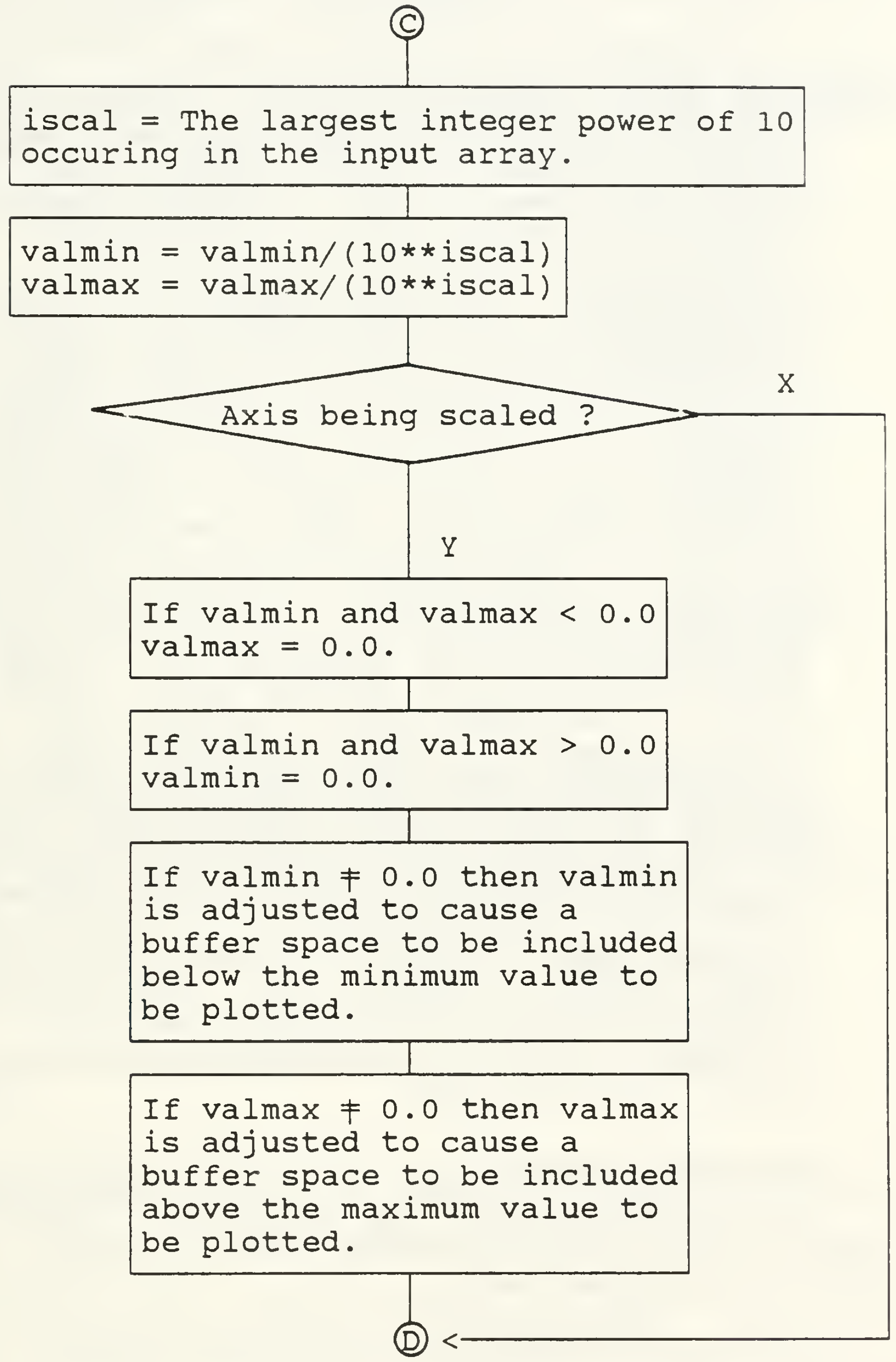

C
iscal = The largest integer power of 10 occuring in the input array.
valmin = valmin/(10**iscal)
valmax = valmax/(10**iscal)
Axis being scaled ?
X
Y
If valmin and valmax < 0.0 valmax = 0.0.
If valmin and valmax > 0.0 valmin = 0.0.
If valmin ≠ 0.0 then valmin is adjusted to cause a buffer space to be included below the minimum value to be plotted.
If valmax ≠ 0.0 then valmax is adjusted to cause a buffer space to be included above the maximum value to be plotted.
D

Figure K.3 SCALE Subroutine Flowchart.

Appendix L

```
C                             DIGFREQ.FOR      VERSION: 2/03/88
C
C
C  PURPOSE:  THIS PROGRAM COMPUTES THE FREQUENCY RESPONSE OF
C            DISCRETE SYSTEMS.  THE PROGRAM CONSISTS OF A MAIN
C            PROGRAM THAT CONTROLS THE INPUT/OUTPUT AND THE
C            SUBROUTINES dfresp AND coeff.  SUBROUTINE dfresp
C            COMPUTES THE FREQUENCY RESPONSE OF EACH SYSTEM.
C            SUBROUTINE coeff ALLOWS THE USER THE OPTION OF
C            GENERATING THE FILTER COEFFICIENTS OF THE SYSTEMS
C            TO BE ANALYZED BY WRITING THE APPROPRIATE EQUATIONS.
C            IF THE USER ELECTS TO GENERATE THE COEFFICIENTS BY
C            USING THE SUBROUTINE coeff, THE EQUATIONS MUST BE
C            WRITTEN INTO THE SUBROUTINE USING STANDARD FORTRAN 77
C            STATEMENTS.  THE COEFFICIENTS MUST BE STORED IN THE
C            ARRAYS b() AND c() WHICH CORRESPOND RESPECTIVELY TO THE
C            NUMERATOR AND DENOMINATOR TERMS OF THE SYSTEM EQUATION.
C            THE USER CAN SELECT ONE OF TWO OPERATING MODES: BATCH
C            OR TEST.  IN BATCH MODE THE AMOUNT OF INTERFACE WITH
C            THE USER IS MINIMIZED AND IT IS ASSUMED THAT THE INPUT
C            DATA HAS BEEN STORED IN THE DEFAULT FILE 'DIGFREQ.IN'.
C            IN TEST MODE THE USER IS PROMPTED FOR THE NAME OF THE
C            INPUT FILE OR HAS THE OPTION TO PERFORM A TRIAL RUN BY
C            USING THE INPUT DATA STORED IN THE FILE 'DIGFREQ.TST'.
C            IT IS RECOMMENDED THAT FIRST-TIME USERS SELECT THE TEST
C            MODE AND MAKE A TRIAL RUN WITH THE PRESTORED INPUT DATA.
C            THE TEST MODE ECHOES THE INPUT DATA ONTO THE MONITOR TO
C            ALLOW VERIFICATION OF ITS ACCURACY.  THIS PROGRAM WILL
C            COMPUTE THE FREQUENCY RESPONSE OF UP TO THREE SYSTEMS.
C            FOR EACH SYSTEM, THE USER HAS THE OPTION OF HAVING THE
C            OUTPUT EXPRESSED IN DECIBELS (dB).  THE OUTPUT OF THIS
C            PROGRAM IS STORED IN TABULAR FORM IN THE FILE
C            'DIGFREQ.OUT' AND IN A FORM SUITABLE FOR PLOTTING
C            IN THE FILE 'DIGFREQ.DAT'.
C
C
C******************************  INPUT  ********************************
C
C  THIS PROGRAM ASSUMES THAT EACH DISCRETE SYSTEM IS MODELED BY THE
C  EQUATION:  H(z) = num/den  WHERE:
C
C  num = b(0)*z**L + b(1)*z**(L-1) + ... + b(L-1)*z + b(L)
C
C  den = c(0)*z**N + c(1)*z**(N-1) + ... + c(N-1)*z + c(N)
C
C    L = A NON-NEGATIVE INTEGER, THE DEGREE OF THE NUMERATOR
C        POLYNOMIAL.
```

```
C    N = A NON-NEGATIVE INTEGER, THE DEGREE OF THE DENOMINATOR
C        POLYNOMIAL.
C    b(0)...b(L) = REAL COEFFICIENTS OF THE NUMERATOR TERMS.
C    c(0)...c(N) = REAL COEFFICIENTS OF THE DENOMINATOR TERMS.
C
C    THE INPUT PARAMETERS SHOULD BE STORED IN A FILE NAMED
C    'DIGFREQ.IN'.  ALL OF THE READ STATEMENTS USED BY THIS PROGRAM
C    REQUIRE FORMATTED INPUT.  PARTICULAR ATTENTION SHOULD BE PAID
C    TO THE FORMATS, ESPECIALLY THE USE OF THE DECIMAL POINT TO
C    DENOTE 'REAL' NUMBERS.  THE INPUT PARAMETERS REQUIRED BY THE
C    PROGRAM ARE LISTED BELOW.
C
C
C  NAME         TYPE       RANGE (ARRAYS)         RESTRICTIONS
C  ----         ----       --------------       ----------------
C  numsys      INTEGER                           1 <= numsys <= 3
C  L           INTEGER                             0 <= L <= 128
C  N           INTEGER                             0 <= N <= 128
C  dsorce      CHARACTER                             'F' OR 'S'
C  yscal       CHARACTER                          'STD' OR 'LOG'
C  theta0      REAL
C  dlthta      REAL
C  numpts      INTEGER                           1 <= numpts <= 101
C  b()         REAL        0, 1, 2, ..., L         0 <= L <= 128
C  c()         REAL        0, 1, 2, ..., N         0 <= N <= 128
C
C  WHERE:
C  numsys = THE NUMBER OF DISTINCT SYSTEMS H(z) TO BE ANALYZED.
C           THIS INTEGER VALUE MUST OCCUR AT THE TOP OF THE INPUT
C           FILE. IT DELINEATES THE NUMBER OF SYSTEMS TO BE READ BY
C           THE PROGRAM AND ANALYZED. FOR EACH SYSTEM (1, ..., numsys)
C           THE PARAMETERS BELOW MUST APPEAR IN THE INPUT FILE.
C
C  L = AN INTEGER VALUE SPECIFYING THE DEGREE OF THE NUMERATOR
C      POLYNOMIAL.
C
C  N = AN INTEGER VALUE SPECIFYING THE DEGREE OF THE DENOMINATOR
C      POLYNOMIAL.
C
C  dsorce = THE CHARACTER STRING 'F' OR 'S' DENOTING WHETHER THE
C           SYSTEM COEFFICIENTS ARE TO BE READ FROM THE INPUT FILE (F)
C           OR GENERATED (S) THROUGH USE OF THE SUBROUTINE coeff.
C
C  yscal  = A CHARACTER STRING SPECIFYING THE DESIRED MAGNITUDE OPTION:
C          'STD' WILL PRODUCE STANDARD MAGNITUDE OUTPUT;
C          'LOG' WILL PRODUCE MAGNITUDE EXPRESSED IN DECIBELS (dB).
C
C  theta0 = THE STARTING VALUE OF THETA (RAD) AS IN Z=EXP(J*THETA).
C
C  dlthta = THE INCREMENT OF THETA (RADIANS).
C
```

```
C  numpts = THE NUMBER OF FREQUENCY POINTS FOR WHICH THE OUTPUT IS
C           TO BE COMPUTED.
C
C  b() =  THE NUMERATOR COEFFICIENTS IN ORDER b(0), b(1), ..., b(L).
C         IF dsorce = 'F' IS SELECTED THEN THE USER MUST SUPPLY THE
C         L+1 NUMERATOR COEFFICIENTS IN THE FILE.  IF dsorce = 'S'
C         THEN THE USER HAS ELECTED TO GENERATE THE NUMERATOR
C         COEFFICIENTS BY WRITING THE APPROPRIATE FORTRAN STATEMENTS
C         IN THE SPACE PROVIDED IN SUBROUTINE coeff.  IF THIS METHOD
C         OF DATA GENERATION IS ELECTED THE PROGRAM MUST BE RECOMPILED
C         BEFORE EXECUTION.
C
C  c() =  THE DENOMINATOR COEFFICIENTS IN ORDER c(0), c(1), ..., c(N).
C         IF dsorce = 'F' IS SELECTED THEN THE USER MUST SUPPLY THE
C         N+1 DENOMINATOR COEFFICIENTS IN THE FILE.  IF dsorce = 'S'
C         THEN THE USER HAS ELECTED TO GENERATE THE DENOMINATOR
C         COEFFICIENTS BY WRITING THE APPROPRIATE FORTRAN STATEMENTS
C         IN THE SPACE PROVIDED IN SUBROUTINE coeff.  IF THIS METHOD
C         OF DATA GENERATION IS ELECTED THE PROGRAM MUST BE RECOMPILED
C         BEFORE EXECUTION.
C
C  NOTE:  THE INPUT FORMAT STATEMENTS OCCUR IN THE MAIN PROGRAM
C         FOLLOWING THE CAPTION: ********  INPUT FORMAT  ********.
C         THE FORM OF THE INPUT DATA FILE IS:
C
C  LINE #                ENTRIES                     FORMAT
C
C  ______               _______                     ______
C    1                   numsys                        i1
C    2              L,N,dsorce,yscal          i3,t11,i3,t21,a1,t31,a3
C    3            dlthta,theta0,numpts               2f10.0,i3
C  NOTE 1           b(k), k=0,1,...,L                 6f10.0
C  NOTE 2           c(k), k=0,1,...,N                 6f10.0
C  NOTE 3
C
C  WHERE:  NN = 1 + (L/6 ROUNDED DOWN TO THE NEXT SMALLER INTEGER).
C          ND = 1 + (N/6 ROUNDED DOWN TO THE NEXT SMALLER INTEGER).
C
C  NOTES 1.  THE NEXT NN LINES ARE ONLY REQUIRED IF dsorce = 'F'. IF
C            dsorce = 'S' THEN THE USER HAS ELECTED TO GENERATE THE
C            L+1 NUMERATOR COEFFICIENTS IN THE SUBROUTINE coeff.
C            THE USER MUST PROVIDE THE APPROPRIATE FORTRAN STATEMENTS
C            IN SUBROUTINE coeff TO GENERATE THE VALUES FOR b().
C
C        2.  THE NEXT ND LINES ARE ONLY REQUIRED IF dsorce = 'F'. IF
C            dsorce = 'S' THEN THE USER HAS ELECTED TO GENERATE THE
C            N+1 DENOMINATOR COEFFICIENTS IN THE SUBROUTINE coeff.
C            THE USER MUST PROVIDE THE APPROPRIATE FORTRAN STATEMENTS
C            IN SUBROUTINE coeff TO GENERATE THE VALUES FOR c().
C
C        3.  FOR numsys > 1 THE FORMAT OF LINES 2... IS REPEATED.
C
```

```
C        4.  THE FORMAT f10.0 USED FOR INPUT DATA PERMITS THE DECIMAL
C            POINT TO BE PLACED ANYWHERE IN THE FIELD OF 10 COLUMNS
C            AND ALSO ALLOWS THE EXPONENTIAL FORMAT TO BE USED (EG.
C            3146.2 = 3.1462E+03).
C
C
C******************************  OUTPUT  ******************************
C
C  THE OUTPUT DATA CREATED BY THE PROGRAM IS STORED IN TABULAR FORM
C  IN THE FILE 'DIGFREQ.OUT'.  ADDITIONALLY, THE OUTPUT DATA IS
C  WRITTEN INTO THE FILE 'DIGFREQ.DAT' TO FACILITATE PLOTTING BY
C  A SEPARATE, USER SUPPLIED PROGRAM.  THE FORMAT OF THE DATA IN
C  'DIGFREQ.DAT' IS:  e12.6, 2x, e12.6.  THE FIRST ENTRY CORRESPONDS
C  TO THE ORDINATE VALUE (THETA) AND THE SECOND ENTRY THE ABSCISSA
C  VALUE (MAGNITUDE OR PHASE).  ADDITIONAL HEADER INFORMATION IS
C  WRITTEN INTO THE DATA FILE TO ALLOW FOR CONTROL AND LABELING OF
C  EACH PLOT.
C
C
C******************************  EXAMPLE  ******************************
C
C  THE INPUT PARAMETERS FOR THE SYSTEM DESCRIBED BELOW ARE STORED IN
C  THE SAMPLE INPUT FILE 'DIGFREQ.TST' AND CAN BE USED FOR A TRIAL
C  RUN IN THE TEST MODE.
C
C
C  SYSTEM:  H(z)=z/(z-0.5)
C
C  GOAL:    TO OBTAIN THE FREQUENCY RESPONSE FOR THIS SYSTEM FROM
C           THETA = 0.0 TO THETA = 3.14159 (PI RADIANS) IN STEPS
C           OF dlthta = PI/10.0
C
C  FOR THE SYSTEM DESCRIBED ABOVE THE INPUT FILE IS:
C
C  1
C  001       001       F         STD
C  0.314159  0.0       011
C  1.0       0.0
C  1.0       -0.5
C
C
C  THE RESULTING OUTPUT DATA FILE: 'DIGFREQ.OUT' IS:
C
C               INPUT DATA FOR SYSTEM #  1
C
C  INPUT DATA SOURCEFILE: DIGFREQ.TST
C  DEGREE OF NUMERATOR =   1
C  DEGREE OF DENOMINATOR =   1
C  dsorce = F
C  NUMBER OF FREQUENCY POINTS =  11     MAGNITUDE OPTION = STD
C  STARTING VALUE OF THETA =  .000000E+00
C  INCREMENT OF THETA =  .314159E+00
```

```
C
C  THE NUMERATOR COEFFICIENTS b(0),b(1)...b(L) ARE
C
C   .1000E+01    .0000E+00
C
C
C  THE DENOMINATOR COEFFICIENTS c(0),c(1)...c(N) ARE
C
C   .1000E+01   -.5000E+00
C
C
C                  OUTPUT DATA FOR SYSTEM #  1
C
C         THETA          MAGNITUDE          PHASE
C       (RADIANS)                         (DEGREES)
C
C      .000000E+00     .200000E+01     .000000E+00
C      .314159E+00     .182897E+01    -.164149E+02
C      .628318E+00     .150588E+01    -.262677E+02
C      .942477E+00     .122886E+01    -.29807E+02
C      .125664E+01     .103088E+01    -.293546E+02
C      .157080E+01     .894428E+00    -.265651E+02
C      .188495E+01     .800894E+00    -.223862E+02
C      .219911E+01     .737654E+00    -.173608E+02
C      .251327E+01     .696900E+00    -.118186E+02
C      .282743E+01     .674038E+00    -.597793E+01
C      .314159E+01     .666667E+00    -.484184E-04
C
C  ------------  END OF RUN, SYSTEM # 1  ------------
C
C
C  FOR ILLUSTRATIVE PURPOSES THE COEFFICIENTS b() AND c() COULD
C  HAVE BEEN GENERATED BY SPECIFYING dsorce = 'S' AND WRITING THE
C  APPROPRIATE FORTRAN STATEMENTS INTO SUBROUTINE coeff.  THE
C  STATEMENTS THAT COULD BE USED TO ACCOMPLISH THIS ARE WRITTEN INTO
C  THE SUBROUTINE BUT ARE 'COMMENTED OUT'.
C
C***************************  MAIN PROGRAM  ***************************


      character infile*12, mode*1, ylabl*13, dsorce*1, yscal*3
      real mh(101), ph(101), thetav(101), c(0:128), b(0:128)

C  PROMPT USER FOR MODE: BATCH OR TEST.

      write(*,1115)
      read(*,1117) mode
      if((mode.eq.'Y').or.(mode.eq.'y')) then
     mode = 'Y'
     write(*,1118)
     read(*,1119) infile
      else
```

```
    infile = 'DIGFREQ.IN'
     endif

C  UNIT=1 DEFINED AS INPUT FILE.  UNITS=2,3 DEFINED AS OUTPUT FILES.

     open(unit=1,file=infile,status='old',iostat=ierr,err=999)
     open(unit=2,file='DIGFREQ.OUT')
     open(unit=3,file='DIGFREQ.DAT')

C  READ INPUT PARAMETERS AND CONDUCT ERROR CHECKS.

     read(1,1000) numsys
     numplts = numsys*2
     write(3,2000) numplts
     if((numsys.lt.1).or.(numsys.gt.3)) then
    write(*,1122) numsys
    stop 'Error, numsys must be in the range: 1 <= numsys <= 3.'
     endif

     do 10 nsys=1, numsys
    data mh/101*0.0/, ph/101*0.0/, thetav/101*0.0/
    data b/129*0.0/, c/129*0.0/

    read(1,1001) L, N, dsorce, yscal
    read(1,1002) dlthta, theta0, numpts

    if((L.lt.0).or.(L.gt.128)) then
      write(*,1124) nsys, L
      stop 'Error, L must be in the range: 0 <= L <= 128.'
    elseif((N.lt.0).or.(N.gt.128)) then
      write(*,1125) nsys, N
      stop 'Error, N must be in the range: 0 <= N <= 128.'
    endif

    if((dsorce.eq.'F').or.(dsorce.eq.'f')) then
      dsorce = 'F'
    elseif((dsorce.eq.'S').or.(dsorce.eq.'s')) then
      dsorce = 'S'
    else
      write(*,1018) dsorce
      stop 'The allowed values for dsorce are: ''S'' or ''F''.'
    endif

    if((numpts.lt.1).or.(numpts.gt.101)) then
      write(*,1127) nsys, numpts
      stop 'Error, numpts must be in the range: 1 <= numpts <= 101.'
    endif

    if((yscal.eq.'STD').or.(yscal.eq.'std')) then
      yscal = 'STD'
      ylabl = '  MAGNITUDE  '
    elseif((yscal.eq.'LOG').or.(yscal.eq.'log')) then
```

```
          yscal = 'LOG'
          ylabl = 'MAGNITUDE(dB)'
        else
          write(*,1128) yscal
          stop 'Error, yscal must be the string: ''LOG'' or ''STD''.'
        endif

C   FOR dsorce = 'F' READ THE COEFFICIENTS b() AND c() FROM THE INPUT
C   FILE.  FOR dsorce = 'S' CALL coeff TO GENERATE THE COEFFICIENTS.

        if(dsorce.eq.'F') then
          read(1,1003) (b(k),k=0,L)
          read(1,1003) (c(k),k=0,N)
        else
          call coeff(L,N,nsys,b,c)
        endif

C   WRITE INPUT DATA INTO THE OUTPUT FILE: DIGFREQ.OUT.

        write(2,1008) nsys
        write(2,1010) infile
        write(2,1110) L
        write(2,1111) N
        write(2,1019) dsorce
        write(2,1112) numpts, yscal
        write(2,1113) theta0
        write(2,1114) dlthta
        write(2,1004)
        write(2,1005) (b(k),k=0,L)
        write(2,1006)
        write(2,1005) (c(k),k=0,N)
        write(2,1009) nsys
        write(2,1126) ylabl
        write(2,1007)

C   FOR TEST MODE ECHO ALL INPUTS ONTO MONITOR (UNIT = *).

        if(mode.eq.'Y') then
          write(*,1120) nsys, infile
          write(*,1110) L
          write(*,1111) N
          write(*,1019) dsorce
          write(*,1112) numpts, yscal
          write(*,1113) theta0
          write(*,1114) dlthta
          write(*,1004)
          write(*,1005) (b(k),k=0,L)
          write(*,1006)
          write(*,1005) (c(k),k=0,N)
          write(*,1123) nsys
          pause 'END OF RUN, STRIKE <CR> WHEN READY TO CONTINUE.'
        endif
```

```
C  CALL dfresp TO COMPUTE THE FREQUENCY RESPONSE.

     call dfresp(b,c,mh,ph,L,N,theta0,dlthta,thetav,numpts,yscal)

C  WRITE RESULTS INTO OUTPUT FILE: DIGFREQ.DAT.

     write(3,2001) numpts
     write(3,*) 'MAGNITUDE RESPONSE'
     write(3,*) 'THETA (rad)'
     write(3,2003) ylabl
     do 55 np=1, numpts
       write(3,2010) thetav(np), mh(np)
55      continue

     write(3,2001) numpts
     write(3,*) 'PHASE RESPONSE'
     write(3,*) 'THETA (rad)'
     write(3,2003) ' PHASE (DEG) '
     do 56 np=1, numpts
       write(3,2010) thetav(np), ph(np)
56      continue

C  WRITE RESULTS INTO OUTPUT FILE: DIGFREQ.OUT.

     do 150 np=1, numpts
       write(2,1013) thetav(np), mh(np), ph(np)
150     continue
     write(2,1123) nsys

10    continue

      write(*,1121)
999   close(unit=1)
      close(unit=2)
      close(unit=3)

      if(ierr.gt.0) then
     write(*,1116) ierr
      endif

C**********  INPUT FORMAT  ***************

1000  format(i1)
1001  format(i3,t11,i3,t21,a1,t31,a3)
1002  format(2f10.0,i3)
1003  format(6f10.0)

C*****************************************

1004  format(t4,'THE NUMERATOR COEFFICIENTS b(0),b(1)...b(L) ARE: ',/)
1005  format(6(2X,e11.4),//)
```

```
1006 0format(//,t4,'THE DENOMINATOR COEFFICIENTS c(0),c(1)...c(N)',
     1' ARE: ',/)
1007  format(t6,'(RADIANS)',t38,'(DEGREES)',/)
1008  format(t16,' INPUT DATA FOR SYSTEM # ',i1,//)
1009  format(///,t16,' OUTPUT DATA FOR SYSTEM # ',i1,/)
1010  format(t4,'INPUT DATA SOURCEFILE: ',a12)
1013  format(t4,3(e12.6,4x))
1018  format(1x,'dsorce = ',a1,2x,'Error, illegal value for dsorce.')
1019  format(t4,'dsorce = ',a1)
1110  format(t4,'DEGREE OF NUMERATOR = ',i3)
1111  format(t4,'DEGREE OF DENOMINATOR = ',i3)
1112 0format(t4,'NUMBER OF FREQUENCY POINTS = ',i3,t39,'MAGNITUDE',
     1' OPTION = ',a3)
1113  format(t4,'STARTING VALUE OF THETA = ',e12.6)
1114  format(t4,'INCREMENT OF THETA = ',e12.6,/)
1115 0format(1x,'DO YOU WISH TO RUN THIS PROGRAM IN TEST',
     1' MODE ?    (Y/N) <CR> : ',\,)
1116 0format(///,1x,'ERROR OPENING INPUT FILE, PROGRAM TERMINATED.',
     1//,1x,'ERROR CODE:',i4,/////)
1117  format(a1)
1118 0format(/////,1x,'TYPE THE NAME OF YOUR DATA FILE FOLLOWED',
     1' BY <CR>.',/,' IF YOU DESIRE TO MAKE A TEST RUN USING THE',
     2' SAMPLE DATA ALREADY STORED',/,' IN THE FILE: DIGFREQ.TST',
     3 '        TYPE: DIGFREQ.TST <CR>',/,' FILENAME: ',\,)
1119  format(a12)
1120  format(////,t4,'SYSTEM # ',i1,'    INPUT DATA SOURCEFILE: ',a12)
1121 0format(//,t4,'TABULAR OUTPUT DATA IS STORED IN FILE: DIGFREQ.OUT',
     1/,t4,'PLOTTING DATA IS STORED IN FILE: DIGFREQ.DAT. ')
1122  format(//////,t2,'The value of numsys is: ',i1,'.')
1123  format(/,1x,13('-'),'  END OF RUN, SYSTEM # ',i1,2x,13('-'),//)
1124 0format(//////,t2,'The degree(L) of the numerator for system ',
     1'# ',i1,' is :  L = ',i3,'.')
1125 0format(//////,t2,'The degree(N) of the denominator for system'
     1,' # ',i1,' is  : N = ',i3,'.')
1126  format(///,t8,'THETA',t21,a13,t40,'PHASE')
1127  format(//////,t2,'The value of numpts for system ',i1,' is: ',i3)
1128  format(/////,t2,'The value of yscal is: ',a3,'.')
2000  format(i1)
2001  format(i3)
2003  format(a13)
2010  format(e12.6,2x,e12.6)

      end


C                          SUBROUTINE: dfresp


C  PURPOSE:  THIS SUBROUTINE COMPUTES THE FREQUENCY RESPONSE OF
C            THE SYSTEM.  ALL FREQUENCY CALCULATIONS ARE IN RADIANS,
C            HOWEVER THE OUTPUT IS CONVERTED TO DEGREES.
```

```
C            THE OUTPUT FORMAT FOR EACH FREQUENCY INCREMENT IS:
C            MAGNITUDE(M)    PHASE(P)      AS IN:  M*EXP(J*P).


      subroutine dfresp(b,c,mh,ph,L,N,theta0,dlthta,thetav,numpts,yscal)
      real mh(numpts), ph(numpts), thetav(numpts), imz, rez
      real b(0:L), c(0:N)
      character yscal*3
      complex z, den, num, h, ci

C  DEFINE CONSTANTS.

      pi = 4.0*atan(1.0)
      ci = (1.0,0.0)

C  ITERATE FROM theta0, IN INCREMENTS OF dlthta.

      do 100 np=1, numpts
     num = ci*b(0)
     den = ci*c(0)
     thetav(np) = theta0 + (np-1)*dlthta
     rez = cos(thetav(np))
     imz = sin(thetav(np))
     z = cmplx(rez,imz)

C  CALCULATE NUMERATOR FOR GIVEN VALUE OF THETA, IF L > 0.

     if(L.gt.0) then
       do 50 k=1, L
         num = z*num + ci*b(k)
 50        continue
     endif

C  CALCULATE DENOMINATOR FOR GIVEN VALUE OF THETA, IF N > 0.

     if(N.gt.0) then
       do 70 k=1, N
         den = z*den + ci*c(k)
 70        continue
     endif
     h = num/den

C  CONVERT COMPLEX VALUE 'h' INTO MAGNITUDE(mh) AND PHASE(ph) TERMS.
C  IF yscal = 'LOG' THEN CONVERT MAGNITUDE TO DECIBELS (dB).
C  DIVIDE BY ZERO AVOIDED BY 'if' STATEMENTS.

     mh(np) = cabs(h)
     if(yscal.eq.'LOG') then
       if(mh(np).gt.0.00001) then
         mh(np) = 20.0*log10(mh(np))
```

```
        else
          mh(np) = -100.0
        endif
      endif

      if(abs(real(h)).lt.1.0e-15) then
        if(abs(aimag(h)).le.1.0e-15) ph(np)=0.0
        if(aimag(h).gt.1.0e-15) ph(np)=90.0
        if(aimag(h).lt.-1.0e-15) ph(np)=-90.0
      else
        ph(np) = (180.0/pi)*atan2(aimag(h),real(h))
      endif

 100  continue

      return
      end



C                      SUBROUTINE: coeff


C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE THE
C            NUMERATOR AND DENOMINATOR COEFFICIENTS THAT DESCRIBE
C            EACH SYSTEM TO BE ANALYZED.  IF dsorce = 'S' THEN
C            THE MAIN PROGRAM WILL CALL THIS SUBROUTINE.


      subroutine coeff(L,N,nsys,b,c)
      real b(0:L), c(0:N)

      pi = 4.0*atan(1.0)
      el = L
      en = N

C***********************************************************************
C  DEVELOP THE EQUATIONS TO GENERATE VALUES FOR THE ARRAYS b() AND c()
C  IN THIS SPACE.  THE STATEMENTS TYPED IN MUST FOLLOW STANDARD
C  FORTRAN 77 RULES AND MAY USE FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS:
C  SIN(), COS(), ABS()...  AN EXAMPLE IS SHOWN BELOW.  NOTE THAT THE
C  VALUE nsys CAN BE USED TO DISTINGUISH BETWEEN SYSTEMS IF MORE THAN
C  ONE SYSTEM (numsys > 1) IS TO BE ANALYZED.
C
C ***  EXAMPLE  ***
C
```

```
C      if(nsys.eq.1) then
C        do 2 i=0, L
C          b(i) = cos(i*pi/(2.0*el))
C 2      continue
C        do 3 i=0, N
C          c(i) = cos(2.0*i*pi/(3.0*en))
C 3      continue
C      endif




C************************************************************************

      return
      end
```

```
C                          ANLGFREQ.FOR               VERSION 2/03/88
C
C
C  PURPOSE:  THIS PROGRAM COMPUTES THE FREQUENCY RESPONSE OF
C            CONTINUOUS-TIME SYSTEMS.  THE PROGRAM CONSISTS OF A
C            MAIN PROGRAM THAT CONTROLS THE INPUT/OUTPUT AND THE
C            SUBROUTINE afresp THAT COMPUTES THE FREQUENCY
C            RESPONSE.  THE USER CAN SELECT ONE OF TWO OPERATING
C            MODES:  BATCH OR TEST.  IN BATCH MODE THE
C            AMOUNT OF INTERFACE WITH THE USER IS MINIMIZED AND
C            IT IS ASSUMED THAT THE INPUT DATA HAS BEEN STORED IN
C            THE DEFAULT FILE 'ANLGFREQ.IN'.  IN THE TEST
C            MODE THE USER IS PROMPTED FOR THE NAME OF THE INPUT
C            FILE OR HAS THE OPTION TO PERFORM A TRIAL RUN BY
C            USING THE INPUT DATA STORED IN THE FILE 'ANLGFREQ.TST'.
C            IT IS RECOMMENDED THAT FIRST-TIME USERS SELECT THE
C            TEST MODE AND MAKE A TRIAL RUN WITH THE PRE-
C            STORED INPUT DATA.  THE TEST MODE ECHOES THE
C            INPUT DATA ONTO THE MONITOR TO ALLOW VERIFICATION OF
C            ITS ACCURACY.  THIS PROGRAM WILL COMPUTE THE FREQUENCY
C            RESPONSE OF UP TO THREE SYSTEMS.  FOR EACH SYSTEM, THE
C            USER HAS THE OPTION OF HAVING THE OUTPUT EXPRESSED IN
C            DECIBELS (dB). THE OUTPUT OF THIS PROGRAM IS STORED IN
C            TABULAR FORM IN THE FILE 'ANLGFREQ.OUT' AND IN A FORM
C            SUITABLE FOR PLOTTING IN THE FILE 'ANLGFREQ.DAT'.
C
C
C****************************   INPUT   *********************************

C
C  THIS PROGRAM ASSUMES THAT EACH CONTINUOUS-TIME SYSTEM IS MODELED
C  BY THE EQUATION:  H(s) = num/den WHERE:
C
C  num = b(0)*s**L + b(1)*s**(L-1) + ... + b(L-1)*s + b(L)
C
C  den = a(0)*s**N + a(1)*s**(N-1) + ... + a(N-1)*s + a(N)
C
C    L = A NON-NEGATIVE INTEGER, THE DEGREE OF THE NUMERATOR
C        POLYNOMIAL.
C    N = A NON-NEGATIVE INTEGER, THE DEGREE OF THE DENOMINATOR
C        POLYNOMIAL.
C    b(0)...b(L) = REAL COEFFICIENTS OF THE NUMERATOR TERMS.
C    a(0)...a(N) = REAL COEFFICIENTS OF THE DENOMINATOR TERMS.
C
C    THE INPUT PARAMETERS SHOULD BE STORED IN A FILE NAMED
C    'ANLGFREQ.IN'.  ALL OF THE READ STATEMENTS USED BY THIS PROGRAM
C    REQUIRE FORMATTED INPUT.  PARTICULAR ATTENTION SHOULD BE PAID
C    TO THE FORMATS, ESPECIALLY THE USE OF THE DECIMAL POINT TO
C    DENOTE 'REAL' NUMBERS.  THE INPUT PARAMETERS REQUIRED BY THE
C    PROGRAM ARE LISTED BELOW.
C
C
```

```
C  NAME         TYPE       RANGE (ARRAYS)          RESTRICTIONS
C  ----         ----       --------------        ----------------
C  numsys       INTEGER                           1 <= numsys <= 3
C  L            INTEGER                            0 <= L <= 128
C  N            INTEGER                            0 <= N <= 128
C  omega0       REAL
C  dlomga       REAL
C  numpts       INTEGER                          1 <= numpts <=101
C  yscal        CHARACTER                          'STD' OR 'LOG'
C  b()          REAL          0,1,2...L            0 <= L <= 128
C  a()          REAL          0,1,2...N            0 <= N <= 128
C
C  WHERE:
C  numsys = THE NUMBER OF DISTINCT SYSTEMS H(s) TO BE ANALYZED.
C           THIS INTEGER VALUE MUST OCCUR AT THE TOP OF THE INPUT
C           FILE. IT DELINEATES THE NUMBER OF SYSTEMS TO BE READ BY
C           THE PROGRAM AND ANALYZED. FOR EACH SYSTEM (1, ..., numsys)
C           THE PARAMETERS BELOW MUST APPEAR IN THE INPUT FILE.
C
C  L = THE DEGREE OF THE NUMERATOR POLYNOMIAL.
C
C  N = THE DEGREE OF THE DENOMINATOR POLYNOMIAL.
C
C  omega0 = THE STARTING VALUE OF OMEGA (RAD/S) AS IN S=J*OMEGA.
C
C  dlomga = THE INCREMENT OF OMEGA (RAD/S).
C
C  numpts = THE NUMBER OF FREQUENCY POINTS FOR WHICH THE OUTPUT IS TO
C           BE COMPUTED.
C
C  yscal = A CHARACTER STRING SPECIFYING THE DESIRED MAGNITUDE OPTION:
C          'STD' WILL PRODUCE STANDARD MAGNITUDE OUTPUT;
C          'LOG' WILL PRODUCE MAGNITUDE EXPRESSED IN DECIBELS (dB).
C
C  b(k) = THE NUMERATOR COEFFICIENTS IN ORDER b(0), b(1), ..., b(L).
C
C  a(k) = THE DENOMINATOR COEFFICIENTS IN ORDER a(0), a(1), ..., a(N).
C
C  NOTE:  THE INPUT FORMAT STATEMENTS OCCUR IN THE MAIN PROGRAM
C         FOLLOWING THE CAPTION: ********  INPUT FORMAT  ********.
C         THE FORM OF THE INPUT DATA FILE IS:
C
C  LINE #                ENTRIES                   FORMAT
C
C  ______                _______                   ______
C    1                   numsys                      i1
C    2              L,N,numpts,yscal         i3,t11,i3,t21,i3,t31,a3
C    3                dlomga,omega0                2f10.0
C  4...4+NN           b(k), k=0,1,...L             6f10.0
C  5+NN...5+NN+N      a(k), k=0,1,...N             6f10.0
C    *
C
```

```
C  WHERE:  NN = L/6 ROUNDED DOWN TO THE NEXT SMALLER INTEGER.
C          ND = N/6 ROUNDED DOWN TO THE NEXT SMALLER INTEGER.
C
C  *NOTE:  FOR numsys > 1 THE FORMAT OF LINES 2... IS REPEATED.
C
C          THE FORMAT f10.0 USED FOR INPUT DATA PERMITS THE DECIMAL
C          POINT TO BE PLACED ANYWHERE IN THE FIELD OF 10 COLUMNS
C          AND ALSO ALLOWS THE EXPONENTIAL FORMAT TO BE USED (EG.
C          3146.2 = 3.1462E+03).
C
C
C*****************************  OUTPUT   ********************************

C
C
C  THE OUTPUT DATA CREATED BY THE PROGRAM IS STORED IN TABULAR FORM
C  IN THE FILE 'ANLGFREQ.OUT'.  ADDITIONALLY, THE OUTPUT DATA IS
C  WRITTEN INTO THE FILE 'ANLGFREQ.DAT' TO FACILITATE PLOTTING BY A
C  SEPARATE, USER SUPPLIED PROGRAM.  THE FORMAT OF THE DATA IN
C  'ANLGFREQ.DAT' IS: e12.6, 2x, e12.6.  THE FIRST ENTRY CORRESPONDS
C  TO THE ORDINATE VALUE (OMEGA) AND THE SECOND ENTRY THE ABSCISSA
C  VALUE (MAGNITUDE OR PHASE).  ADDITIONAL HEADER INFORMATION IS
C  WRITTEN INTO THE DATA FILE TO ALLOW FOR CONTROL AND LABELING OF
C  EACH PLOT.
C
C
C****************************  EXAMPLE  *********************************

C
C
C  THE INPUT PARAMETERS FOR THE SYSTEM DESCRIBED BELOW ARE STORED IN
C  THE SAMPLE INPUT FILE 'ANLGFREQ.TST' AND CAN BE USED FOR A TRIAL
C  RUN IN THE TEST MODE.
C
C
C  SYSTEM:  H(s) = 10.0*s/(s**2 + 6.0*s + 5.0)
C
C  GOAL:     TO OBTAIN THE FREQUENCY RESPONSE FOR THIS SYSTEM FROM
C            OMEGA = 0.0 TO OMEGA = 4.0 (RAD/S) IN STEPS OF
C            dlomga = 0.2 (RAD/S).
C
C  FOR THE SYSTEM DESCRIBED ABOVE THE INPUT FILE IS:
C
C  001
C  001       002       021       STD
C  0.2       0.0
C  10.0      0.0
C  1.0       6.0       5.0
C
C
```

```
C  THE RESULTING OUTPUT DATA FILE ('ANLGFREQ.OUT') IS:
C
C                 INPUT DATA FOR SYSTEM #  1
C
C  INPUT DATA SOURCEFILE: ANLGFREQ.TST
C  DEGREE OF NUMERATOR =   1
C  DEGREE OF DENOMINATOR =   2
C  NUMBER OF FREQUENCY POINTS =   21       MAGNITUDE OPTION =  STD
C  STARTING VALUE OF OMEGA =  .000000E+00
C  INCREMENT OF OMEGA =  .200000E+00
C
C  THE NUMERATOR COEFFICIENTS b(0),b(1)...b(L) ARE:
C
C   .1000E+02    .0000E+00
C
C
C  THE DENOMINATOR COEFFICIENTS a(0),a(1)...a(N) ARE:
C
C   .1000E+01    .6000E+01    .5000E+01
C
C
C                 OUTPUT DATA FOR SYSTEM #   1
C
C       OMEGA          MAGNITUDE          PHASE
C      (rad/s)                          (DEGREES)
C
C    .000000E+00     .000000E+00     .000000E+00
C    .200000E+00     .391919E+00     .763995E+02
C    .400000E+00     .740416E+00     .636247E+02
C    .600000E+00     .102166E+01     .521935E+02
C    .800000E+00     .123370E+01     .422499E+02
C    .100000E+01     .138675E+01     .336901E+02
C    .120000E+01     .149402E+01     .263098E+02
C    .140000E+01     .156719E+01     .198954E+02
C    .160000E+01     .161531E+01     .142607E+02
C    .180000E+01     .164497E+01     .925573E+01
C    .200000E+01     .166091E+01     .476364E+01
C    .220000E+01     .166654E+01     .694459E+00
C    .240000E+01     .166435E+01    -.302114E+01
C    .260000E+01     .165616E+01    -.643692E+01
C    .280000E+01     .164335E+01    -.959500E+01
C    .300000E+01     .162698E+01    -.125288E+02
C    .320000E+01     .160786E+01    -.152652E+02
C    .340000E+01     .158665E+01    -.178262E+02
C    .360000E+01     .156386E+01    -.202298E+02
C    .380000E+01     .153990E+01    -.224913E+02
C    .400000E+01     .151511E+01    -.246236E+02
C
C  -------------  END OF RUN, SYSTEM # 1  ------------
C
C
```

```
C*************************  MAIN PROGRAM  ****************************

      character infile*12, mode*1, ylabl*13, yscal*3
      real mh(101), ph(101), omegav(101), a(0:128), b(0:128)

C  PROMPT USER FOR MODE: BATCH OR TEST.

      write(*,1115)
      read(*,1117) mode
      if((mode.eq.'Y').or.(mode.eq.'y')) then
     mode = 'Y'
     write(*,1118)
     read(*,1119) infile
      else
     infile = 'ANLGFREQ.IN'
      endif

C  UNIT=1 DEFINED AS INPUT FILE.  UNITS=2,3 DEFINED AS OUTPUT FILES.

      open(unit=1,file=infile,status='old',iostat=ierr,err=999)
      open(unit=2,file='ANLGFREQ.OUT')
      open(unit=3,file='ANLGFREQ.DAT')

C  READ INPUT PARAMETERS AND CONDUCT ERROR CHECKS.

      read(1,1000) numsys
      numplts = numsys*2
      write(3,2000) numplts

      if((numsys.lt.1).or.(numsys.gt.3)) then
     write(*,1122) numsys
     stop 'Error, numsys must be in the range: 1 <= numsys <= 3.'
      endif

      do 10 nsys=1, numsys

     data mh/101*0.0/, ph/101*0.0/, omegav/101*0.0/
     data a/129*0.0/, b/129*0.0/

     read(1,1001) L, N, numpts, yscal
     read(1,1002) dlomga, omega0

     if((L.lt.0).or.(L.gt.128)) then
       write(*,1124) nsys, L
       stop 'Error, L must be in the range: 0 <= L <= 128.'
     elseif((N.lt.0).or.(N.gt.128)) then
       write(*,1125) nsys, N
       stop 'Error, N must be in the range: 0 <= N <= 128.'
     endif
```

```
      if((numpts.lt.1).or.(numpts.gt.101)) then
        write(*,1127) nsys, numpts
        stop 'Error, numpts must be in the range: 0 <= numpts <= 101.'
      endif

      if((yscal.eq.'STD').or.(yscal.eq.'std')) then
        yscal = 'STD'
        ylabl = '  MAGNITUDE  '
      elseif((yscal.eq.'LOG').or.(yscal.eq.'log')) then
        yscal = 'LOG'
        ylabl = 'MAGNITUDE(dB)'
      else
        write(*,1128) yscal
        stop 'Error, yscal must be the string: ''LOG'' or ''STD''.'
      endif

C  READ THE SYSTEM COEFFICIENTS b() AND a().

      read(1,1003) (b(k),k=0,L)
      read(1,1003) (a(k),k=0,N)

C  WRITE THE INPUT PARAMETERS INTO OUTPUT FILE: ANLGFREQ.OUT.

      write(2,1008) nsys
      write(2,1010) infile
      write(2,1110) L
      write(2,1111) N
      write(2,1112) numpts, yscal
      write(2,1113) omega0
      write(2,1114) dlomga
      write(2,1004)
      write(2,1005) (b(k),k=0,L)
      write(2,1006)
      write(2,1005) (a(k),k=0,N)
      write(2,1009) nsys
      write(2,1126) ylabl
      write(2,1007)

C  FOR TEST MODE ECHO ALL INPUTS ONTO MONITOR (UNIT = *).

      if(mode.eq.'Y') then
        write(*,1120) nsys, infile
        write(*,1110) L
        write(*,1111) N
        write(*,1112) numpts, yscal
        write(*,1113) omega0
        write(*,1114) dlomga
        write(*,1004)
        write(*,1005) (b(k),k=0,L)
        write(*,1006)
        write(*,1005) (a(k),k=0,N)
        write(*,1123) nsys
        pause 'END OF RUN, STRIKE <CR> WHEN READY TO CONTINUE'
      endif

C   CALL afresp TO COMPUTE FREQUENCY RESPONSE.

      call afresp(b,a,mh,ph,L,N,omega0,dlomga,omegav,numpts,yscal)

C   WRITE RESULTS INTO OUTPUT FILE: ANLGFREQ.DAT.

      write(3,2001) numpts
      write(3,*) 'MAGNITUDE RESPONSE'
      write(3,*) 'OMEGA(rad/s)'
      write(3,2003) ylabl
      do 55 np = 1, numpts
        write(3,2010) omegav(np), mh(np)
55       continue

      write(3,2001) numpts
      write(3,*) 'PHASE RESPONSE'
      write(3,*) 'OMEGA(rad/s)'
      write(3,2003) ' PHASE (DEG) '
      do 56 np = 1, numpts
        write(3,2010) omegav(np), ph(np)
56       continue

C   WRITE RESULTS INTO OUTPUT FILE: ANLGFREQ.OUT.

      do 150 np=1, numpts
        write(2,1013) omegav(np), mh(np), ph(np)
150      continue
      write(2,1123) nsys

10     continue

       write(*,1121)
999    close(unit=1)
       close(unit=2)
       close(unit=3)

       if(ierr.gt.0) then
      write(*,1116) ierr
       endif

C  ************  INPUT FORMAT  *************

1000   format(i3)
1001   format(i3,t11,i3,t21,i3,t31,a3)
1002   format(2f10.0)
1003   format(6(f10.0))
```

```
C  *******************************************

1004  format(t4,'THE NUMERATOR COEFFICIENTS b(0),b(1)...b(L) ARE:',/)
1005  format(6(2X,e11.4),//)
1006 0format(//,t4,'THE DENOMINATOR COEFFICIENTS a(0),a(1)...a(N)',
     1' ARE:',/)
1007  format(t8,'(rad/s)',24x,'(DEGREES)',/)
1008  format(t16,' INPUT DATA FOR SYSTEM # ',i1,//)
1009  format(///,t16,' OUTPUT DATA FOR SYSTEM # ',i1,/)
1010  format(t4,'INPUT DATA SOURCEFILE: ',a12)
1013  format(t4,3(e12.6,4x))
1110  format(t4,'DEGREE OF NUMERATOR = ',i3)
1111  format(t4,'DEGREE OF DENOMINATOR = ',i3)
1112 0format(t4,'NUMBER OF FREQUENCY POINTS = ',i3,t40,'MAGNITUDE',
     1' OPTION = ',a3)
1113  format(t4,'STARTING VALUE OF OMEGA = ',e12.6)
1114  format(t4,'INCREMENT OF OMEGA = ',e12.6,/)
1115 0format(1x,'DO YOU WISH TO RUN THIS PROGRAM IN TEST',
     1' MODE ?    (Y/N) <CR> : ',\,)
1116 0format(///,1x,'ERROR OPENING INPUT FILE, PROGRAM TERMINATED.',
     1//,1x,'ERROR CODE:',i4,/////)
1117  format(a1)
1118 0format(/////,1x,'TYPE THE NAME OF YOUR DATA FILE FOLLOWED',
     1' BY <CR>.',/,' IF YOU DESIRE TO MAKE A TEST RUN USING THE',
     2' SAMPLE DATA ALREADY STORED',/,' IN THE FILE: ANLGFREQ.TST',
     3 '       TYPE: ANLGFREQ.TST <CR>',/,' FILENAME: ',\,)
1119  format(a12)
1120  format(////,t4,'SYSTEM # ',i1,'   INPUT DATA SOURCEFILE: ',a12)
1121 0format(/,t4,'TABULAR OUTPUT DATA IS STORED IN FILE: ANLGFREQ.OUT'
     1,/,t4,'PLOTTING DATA IS STORED IN FILE: ANLGFREQ.DAT')
1122  format(//////,t2,'The value of numsys is: ',i1,'.')
1123  format(/,1x,13('-'),'  END OF RUN, SYSTEM #',i1,2x,13('-'),//)
1124 0format(//////,t2,'The degree(L) of the numerator for system ',
     1'# ',i1,' is : L = ',i3,'.')
1125 0format(//////,t2,'The degree(N) of the denominator for system ',
     1'# ',i1,' is : N = ',i3,'.')
1126  format(///,t8,'OMEGA',t21,a13,t40,'PHASE')
1127  format(//////,t2,'The value of numpts for system ',i1,' is: ',i3)
1128  format(//////,t2,'The value of yscal is: ',a3,'.')
2000  format(i1)
2001  format(i3)
2003  format(a13)
2010  format(e12.6,2x,e12.6)

      end
```

```
C                    SUBROUTINE: afresp


C  PURPOSE:  THIS SUBROUTINE COMPUTES THE FREQUENCY RESPONSE OF
C            THE SYSTEM.  ALL FREQUENCY CALCULATIONS ARE IN RADIANS,
C            HOWEVER THE OUTPUT IS CONVERTED TO DEGREES.
C            THE OUTPUT FORMAT FOR EACH FREQUENCY INCREMENT IS:
C            MAGNITUDE(M)    PHASE(P)       AS IN:   M*EXP(J*P).


      subroutine  afresp(b,a,mh,ph,L,N,omega0,dlomga,omegav,numpts,yscal)

      real mh(numpts), ph(numpts), omegav(numpts)
      real b(0:L), a(0:N), ims, res
      character yscal*3
      complex s, den, num, h, ci

C  DEFINE CONSTANTS.

      ci = (1.0,0.0)
      pi = 4.0*atan(1.0)

C  ITERATE FROM omega0, IN INCREMENTS OF dlomga.

      do 100 np=1, numpts
     num = ci*b(0)
     den = ci*a(0)
     omegav(np) = omega0 + (np-1)*dlomga
     res = 0.0
     ims = omegav(np)
     s = cmplx(res,ims)

C  CALCULATE NUMERATOR FOR GIVEN VALUE OF OMEGA, IF L > 0.

     if(L.gt.0) then
       do 50 k=1, L
         num = s*num + ci*b(k)
50        continue
     endif

C  CALCULATE DENOMINATOR FOR GIVEN VALUE OF OMEGA, IF N > 0.

     if(N.gt.0) then
       do 70 k=1, N
         den = s*den + ci*a(k)
70        continue
     endif

     h = num/den
```

```
C  CONVERT COMPLEX VALUE 'h' INTO MAGNITUDE(mh) AND PHASE(ph) TERMS.
C  IF yscal = 'LOG' THEN CONVERT MAGNITUDE TO DECIBELS (dB).
C  DIVIDE BY ZERO AVOIDED BY 'if' STATEMENTS.

     mh(np) = cabs(h)
     if(yscal.eq.'LOG') then
       if(mh(np).gt.0.00001) then
         mh(np) = 20.0*log10(mh(np))
       else
         mh(np) = -100.0
       endif
     endif

     if(abs(real(h)).lt.1.0e-15) then
       if(abs(aimag(h)).le.1.0e-15) ph(np)=0.0
       if(aimag(h).gt.1.0e-15) ph(np)= 90.0
       if(aimag(h).lt.-1.0e-15) ph(np)=-90.0
     else
       ph(np) = (180.0/pi)*atan2(aimag(h),real(h))
     endif

100   continue

      return
      end
```

```
C                               DFT.FOR          VERSION: 2/03/88
C
C
C  PURPOSE:  THIS PROGRAM COMPUTES THE DISCRETE FOURIER TRANSFORM
C            (DFT) OR THE INVERSE DISCRETE FOURIER TRANSFORM (IDFT) OF
C            A SEQUENCE OF COMPLEX INPUT DATA.  THE PROGRAM CONSISTS
C            OF A MAIN PROGRAM AND THREE SUBROUTINES. THE SUBROUTINE
C            dft COMPUTES THE DISCRETE FOURIER TRANSFORM OF THE
C            INPUT ARRAY; THE SUBROUTINE invdft COMPUTES THE
C            INVERSE DISCRETE FOURIER TRANSFORM; AND THE SUBROUTINE
C            sample ALLOWS THE USER TO GENERATE THE INPUT SEQUENCE BY
C            WRITING THE APPROPRIATE EQUATIONS.  IF THE USER ELECTS
C            TO GENERATE THE INPUT DATA BY USING THE SUBROUTINE
C            sample, THE EQUATIONS MUST BE WRITTEN INTO THE
C            SUBROUTINE USING STANDARD FORTRAN 77 EXECUTABLE STATE-
C            MENTS AND THE INPUT DATA GENERATED MUST BE STORED IN
C            THE ARRAY xin(). THE OUTPUT OF 'DFT.FOR' IS STORED IN
C            THE ARRAY xout(). THE USER HAS THE OPTION OF SELECTING
C            ONE OF TWO OPERATING MODES: BATCH OR TEST. IN BATCH
C            MODE THE AMOUNT OF INTERACTION WITH THE USER IS
C            MINIMIZED AND IT IS ASSUMED THAT THE INPUT PARAMETERS
C            HAVE BEEN STORED IN THE INPUT FILE 'DFT.IN'.  IN
C            TEST MODE THE USER IS PROMPTED FOR THE NAME OF THE
C            INPUT FILE OR HAS THE OPTION TO PERFORM A TRIAL
C            RUN USING THE DATA STORED IN THE FILE 'DFT.TST'.
C            IT IS RECOMMENDED THAT FIRST-TIME USERS SELECT TEST
C            MODE AND MAKE A TRIAL RUN WITH THE PRESTORED
C            DATA.  THE TEST MODE ECHOES PORTIONS OF THE
C            INPUT DATA ONTO THE MONITOR TO ALLOW VERIFICATION
C            OF ITS ACCURACY.  THE OUTPUT IS STORED IN TABULAR FORM
C            IN THE FILE 'DFT.OUT' AND IN A FORM SUITABLE FOR
C            PLOTTING IN THE FILE 'DFT.DAT'.
C
C
C*****************************  INPUT  ********************************

C
C
C  THIS PROGRAM ASSUMES THAT THERE ARE 'N' COMPLEX VALUES IN THE
C  INPUT SEQUENCE.  THE INPUT SEQUENCE IS ASSUMED TO BE DEFINED IN THE
C  INTERVAL: 0 TO N-1. IF THE INPUT SEQUENCE CONSISTS OF 'REAL' NUMBERS,

C  THE IMAGINARY PART IS STORED AS 0.0.  THE VALUE 'N' AS WELL AS THE
C  OTHER PARAMETERS DESCRIBED BELOW SHOULD BE STORED IN THE INPUT
C  FILE 'DFT.IN'.  ALL OF THE READ STATEMENTS USED BY THIS PROGRAM
C  REQUIRE FORMATTED INPUT.  PARTICULAR ATTENTION SHOULD BE PAID TO
C  THESE FORMATS, ESPECIALLY THE USE OF THE DECIMAL POINT TO
C  DISTINGUISH BETWEEN 'REAL' AND INTEGER DATA.
C
C
```

```
C  NAME         TYPE       RANGE (ARRAYS)          RESTRICTIONS
C  ----         ----       --------------         ---------------
C  N            INTEGER                             1 <= N <= 256
C  dsorce       CHARACTER                            'F'  OR  'S'
C  option       CHARACTER                           'DFT' OR 'INV'
C  xin()        COMPLEX     0, 1, ..., N-1          1 <= N <= 256
C
C  WHERE:
C
C  N = AN INTEGER THAT SPECIFIES THE NUMBER OF COMPLEX VALUES IN THE
C      INPUT SEQUENCE.
C
C  dsorce = A CHARACTER VALUE OF 'F' OR 'S' DENOTING WHETHER THE
C           INPUT SEQUENCE IS TO BE READ FROM A FILE (F) OR TO BE
C           GENERATED (S) BY A USER-DEFINED EQUATION LOCATED IN THE
C           SUBROUTINE sample.
C
C  option = A CHARACTER STRING OF THE LETTERS 'DFT' OR 'INV'
C           DENOTING WHETHER THE DFT OR THE INVERSE DFT IS TO BE
C           PERFORMED ON THE INPUT DATA.
C
C  xin() = THE ARRAY OF COMPLEX INPUT DATA.  IF dsorce = 'F'
C          IS SELECTED THEN THE USER MUST SUPPLY THE N INPUT
C          VALUES IN THE FILE.  IF dsorce = 'S' THEN THE USER
C          HAS ELECTED TO GENERATE THE INPUT SEQUENCE BY WRITING
C          THE APPROPRIATE FORTRAN STATEMENTS IN THE SPACE
C          PROVIDED IN SUBROUTINE sample.  IF THIS METHOD
C          OF DATA GENERATION IS ELECTED THE PROGRAM MUST BE
C          RECOMPILED BEFORE EXECUTION.
C
C  NOTE:  THE INPUT FORMAT STATEMENTS OCCUR IN THE MAIN PROGRAM
C         FOLLOWING THE CAPTION: ********  INPUT FORMAT  ********.
C         THE FORM OF THE INPUT DATA FILE IS:
C
C  LINE#                   ENTRIES                     FORMAT
C  -----                   -------                     ------
C    1                  N,dsorce,option           i3,t11,a1,t21,a3
C  2...N+1                 xin()                       2f10.0
C
C
C  NOTES 1.  LINES 2...N+1 ARE ONLY REQUIRED IF dsorce = 'F'.
C            IF dsorce = 'S' THEN THE USER HAS ELECTED TO GENERATE
C            THE N VALUES FOR xin() IN THE SUBROUTINE sample.  THE
C            USER MUST PROVIDE THE APPROPRIATE FORTRAN STATEMENTS IN
C            SUBROUTINE sample TO GENERATE xin().
C
C        2.  THE FORMAT f10.0 USED FOR INPUT DATA PERMITS THE
C            DECIMAL POINT TO BE PLACED ANYWHERE IN THE FIELD OF TEN
C            COLUMNS AND ALSO ALLOWS THE EXPONENTIAL FORMAT TO BE
C            USED (E.G., 3146.2 = 3.1462E+03).
C
C
```

```
C*****************************  OUTPUT  ******************************
C
C
C  THE INPUT DATA AS WELL AS THE OUTPUT DATA ARE STORED IN TABULAR FORM

C  IN THE FILE 'DFT.OUT'.  ADDITIONALLY, THE INPUT SEQUENCE (REAL AND
C  IMAGINARY) AND THE OUTPUT SEQUENCE (MAGNITUDE AND PHASE) ARE WRITTEN

C  INTO THE FILE 'DFT.DAT' TO FACILITATE PLOTTING BY A SEPARATE, USER
C  SUPPLIED PROGRAM.  THE FORMAT OF THE DATA IN 'DFT.DAT' IS:
C  e12.6, 2x, e12.6.  THE FIRST ENTRY CORRESPONDS TO THE ORDINATE VALUE

C  AND THE SECOND ENTRY, THE ABSCISSA VALUE.  ADDITIONAL HEADER
C  INFORMATION IS WRITTEN INTO 'DFT.DAT' TO ALLOW FOR CONTROL AND
C  LABELING OF EACH PLOT.
C
C
C*****************************  EXAMPLE  *****************************
C
C
C  THE INPUT PARAMETERS BELOW ARE STORED IN THE INPUT FILE
C  'DFT.TST'.  THERE ARE FIVE DATA POINTS IN THE INPUT SEQUENCE AND THE

C  GOAL IS TO CALCULATE THE DISCRETE FOURIER TRANSFORM OF THE SEQUENCE.

C
C
C  005        F         DFT
C  0.0        0.0
C  1.0        0.0
C  2.0        0.0
C  3.0        0.0
C  4.0        0.0
C
C
C  THE RESULTING OUTPUT DATA FILE 'DFT.OUT' IS:
C
C  INPUT DATA SOURCEFILE: DFT.TST
C  VALUE OF N =   5     dsorce = F     option = DFT
C
C
C                INPUT DATA
C
C   SAMPLE #   REAL          IMAGINARY
C
C       0     .000000E+00   .000000E+00
C       1     .100000E+01   .000000E+00
C       2     .200000E+01   .000000E+00
C       3     .300000E+01   .000000E+00
C       4     .400000E+01   .000000E+00
```

```
C
C
C                                  OUTPUT DATA
C
C   SAMPLE #   REAL           IMAGINARY      MAGNITUDE        PHASE
C                                                           (DEGREES)
C      0       .100000E+02    .000000E+00    .100000E+02    .000000E+00
C      1      -.250000E+01    .344096E+01    .425325E+01    .126000E+03
C      2      -.250000E+01    .812300E+00    .262866E+01    .162000E+03
C      3      -.250000E+01   -.812299E+00    .262866E+01   -.162000E+03
C      4      -.250000E+01   -.344096E+01    .425326E+01   -.126000E+03
C
C
C  FOR ILLUSTRATIVE PURPOSES THE INPUT SEQUENCE xin() COULD HAVE BEEN
C  GENERATED BY SPECIFYING dsorce = 'S' AND WRITING THE APPROPRIATE
C  FORTRAN STATEMENTS INTO SUBROUTINE sample.  THE STATEMENTS THAT
C  COULD BE USED TO ACCOMPLISH THIS ARE WRITTEN INTO THE SUBROUTINE
C  BUT ARE 'COMMENTED OUT'.
C
C
C**************************  MAIN PROGRAM  ****************************

      character infile*12, option*3, mode*1, dsorce*1, yscal*3
      complex xin(0:255), xout(0:255)
      real xmag(0:255), xph(0:255), nn

C  PROMPT USER FOR MODE: BATCH OR TEST.

      write(*,1115)
      read(*,1117) mode
      if((mode.eq.'y').or.(mode.eq.'Y')) then
     mode = 'Y'
     write(*,1118)
     read(*,1119) infile
      else
     infile= 'DFT.IN'
      endif

C  UNIT=1 DEFINED AS INPUT FILE.  UNITS=2,3 DEFINED AS OUTPUT FILES.

      open(unit=1,file=infile,status='old',iostat=ierr,err=999)
      open(unit=2,file='DFT.OUT')
      open(unit=3,file='DFT.DAT')

C  READ INPUT PARAMETERS AND CONDUCT ERROR CHECKS.

      read(1,1000) N, dsorce, option
```

```
       if((N.lt.1).or.(N.gt.256)) then
      write(*,1010) N
      stop 'The allowed values for N are:  1 <= N <= 256.'
       endif

       if((option.eq.'dft').or.(option.eq.'DFT')) then
      option = 'DFT'
       elseif((option.eq.'inv').or.(option.eq.'INV')) then
      option = 'INV'
       else
      write(*,1011) option
      stop 'The allowed values for option are: ''DFT'' or ''INV''.'
       endif

       if((dsorce.eq.'f').or.(dsorce.eq.'F')) then
      dsorce = 'F'
       elseif((dsorce.eq.'s').or.(dsorce.eq.'S')) then
      dsorce = 'S'
       else
      write(*,1009) dsorce
      stop 'The allowed values for dsorce are: ''S'' or ''F''.'
       endif

C   DEFINE CONSTANTS.

       en = N
       k = 8
       pi = 4.0*atan(1.0)
       numplts = 4
       yscal = 'STD'

C   FOR dsorce = 'F' READ THE INPUT SEQUENCE FROM THE INPUT FILE.
C   FOR dsorce = 'S' CALL sample TO GENERATE THE INPUT SEQUENCE.
C   THE INPUT SEQUENCE IS STORED IN THE ARRAY xin().

       if(dsorce.eq.'F') then
      read(1,1001) (xin(i),i=0,N-1)
       else
      call sample(xin,N)
       endif

C   FOR TEST MODE ECHO INPUT DATA ONTO THE MONITOR (UNIT = *).

       if(mode.eq.'Y') then
      write(*,1016) infile
      if(N.lt.8) k=N
      write(*,1017) N, dsorce, option
```

```
    write(*,1012) k
    write(*,1015)
    do 1 i=0, k-1
      write(*,1020) i, xin(i)
1       continue
     endif

C  WRITE THE INPUT SEQUENCE INTO FILE: DFT.DAT.

     write(3,2000) numplts
     write(3,2001) N
     write(3,*) 'INPUT SEQUENCE (REAL)'
     write(3,*) 'SAMPLE # '
     write(3,*) 'REAL xin()'
     do 55 i=0, N-1
    nn = i
    write(3,2010) nn, real(xin(i))
55   continue

     write(3,2001) N
     write(3,*) 'INPUT SEQUENCE (IMAGINARY)'
     write(3,*) 'SAMPLE # '
     write(3,*) 'IMAG xin()'
     do 56 i=0, N-1
    nn = i
    write(3,2010) nn, aimag(xin(i))
56   continue

C  WRITE INPUT DATA INTO FILE: DFT.OUT.

     write(2,1016) infile
     write(2,1017) N, dsorce, option
     write(2,1014)
     write(2,1015)
     do 2 i=0, N-1
    write(2,1020) i, xin(i)
2    continue

C  CALL dft OR invdft TO PERFORM THE SELECTED COMPUTATION.

     if(option.eq.'INV') then
    call invdft(N,xin,xout)
     else
    call dft(N,xin,xout)
     endif

C  TRANSFORM OUTPUT DATA INTO EXPONENTIAL FORM: xmag*EXP(j*xph).
C  PHASE xph() IS EXPRESSED IN DEGREES.

     do 60 i=0, N-1
    xmag(i) = cabs(xout(i))
    if(abs(real(xout(i))).lt.1.0e-15) then
```

```
          if(abs(aimag(xout(i))).le.1.0e-15) xph(i)=0.0
          if(aimag(xout(i)).gt.1.0e-15) xph(i)=90.0
          if(aimag(xout(i)).lt.-1.0e-15) xph(i)=-90.0
        else
          xph(i) = (180.0/pi)*atan2(aimag(xout(i)),real(xout(i)))
        endif
60       continue

C  WRITE THE OUTPUT DATA INTO FILE: DFT.DAT.

         write(3,2001) N
         write(3,*) 'OUTPUT MAGNITUDE'
         write(3,*) 'SAMPLE #'
         write(3,*) 'MAGNITUDE'
         do 57 i=0, N-1
        nn = i
        write(3,2010) nn, xmag(i)
57       continue

         write(3,2001) N
         write(3,*) 'OUTPUT PHASE'
         write(3,*) 'SAMPLE #'
         write(3,*) 'PHASE (DEG)'
         do 58 i=0, N-1
        nn = i
        write(3,2010) nn, xph(i)
58       continue

C  WRITE THE OUTPUT DATA INTO FILE: DFT.OUT.

         write(2,1025)
         do 5 i=0, N-1
        write(2,1030) i, xout(i), xmag(i), xph(i)
5        continue

         write(*,1019)
999      close(unit=1)
         close(unit=2)
         close(unit=3)

         if(ierr.gt.0) then
        write(*,1116) infile, ierr
         endif

C  ********  INPUT FORMAT  ********

1000     format(i3,t11,a1,t21,a3)
1001     format(2f10.0)
```

```
C  *******************************

1009  format(1x,'dsorce = ',a1,2x,'Error, illegal value for dsorce.')
1010  format(1x,'N = ',i3,2x,'Error, value of N not allowed.')
1011  format(1x,'option = ',a3,2x,'Error, illegal value for option.')
1012 0format(/,' THE FIRST ',i1,' VALUES OF xin() ARE LISTED BELOW.',
     1/,' VERIFY THAT THE DATA IS CORRECT.',/)
1014  format(//,t19,'INPUT DATA',//)
1015  format(/,t4,'SAMPLE #',t15,'REAL',t29,'IMAGINARY',/)
1016  format(//////,' INPUT DATA SOURCEFILE: ',a12)
1017  format(' VALUE OF N = ',i3,5x,'dsorce = ',a1,5x,'option = ',a3)
1019 0format(//,' TABULAR OUTPUT DATA IS STORED IN FILE: DFT.OUT.',
     1/,' PLOTTING DATA IS STORED IN FILE: DFT.DAT.')
1020  format(t6,i3,t13,2(e12.6,2x))
1025 0format(///,t33,'OUTPUT DATA',//,t4,'SAMPLE #',t17,'REAL',
     1t33,'IMAGINARY',t49,'MAGNITUDE',t67,'PHASE',/,t65,'(DEGREES)')
1030  format(t5,i3,t15,4(e12.6,4x))
1115 0format(1x,'DO YOU WISH TO RUN THIS PROGRAM IN TEST',
     1' MODE ?    (Y/N) <CR> : ',\,)
1116 0format(///,1x,'ERROR OPENING INPUT FILE: ',a12,/,1x,'PROGRAM',
     1' TERMINATED.',//,1x,'ERROR CODE:',i4,/////)
1117  format(a1)
1118 0format(/////,1x,'TYPE THE NAME OF YOUR DATA FILE FOLLOWED',
     1' BY <CR>.',/,' IF YOU DESIRE TO MAKE A TEST RUN USING THE',
     2' SAMPLE DATA ALREADY STORED',/,' IN THE FILE: DFT.TST',
     3'        TYPE: DFT.TST <CR>',/,' FILENAME: ',\,)
1119  format(a12)
2000  format(i1)
2001  format(i3)
2010  format(e12.6,2x,e12.6)

      end


C                        SUBROUTINE: invdft


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY
C            xin(), COMPUTES THE INVERSE DFT OF THE ARRAY, AND
C            RETURNS THE RESULTS IN THE ARRAY xout().


      subroutine invdft(N,xin,xout)
      complex xin(0:N-1), xout(0:N-1)

      en = N
```

```
C  COMPUTE THE COMPLEX CONJUGATE OF THE INPUT SEQUENCE.

      do 70 i=0, N-1
     xin(i) = conjg(xin(i))
70    continue

C  COMPUTE THE DISCRETE FOURIER TRANSFORM OF THE ARRAY.

      call dft(N,xin,xout)

C  COMPUTE THE COMPLEX CONJUGATE OF THE RESULTING ARRAY.

      do 80 i=0, N-1
     xout(i) = conjg(xout(i))/en
80    continue

      return
      end



C                       SUBROUTINE: dft


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY
C            xin(), COMPUTES THE DISCRETE FOURIER TRANSFORM (DFT)
C            OF THE ARRAY, AND RETURNS THE RESULTING SEQUENCE IN THE
C            COMPLEX ARRAY xout().


      subroutine dft(N,xin,xout)
      complex xin(0:N-1), xout(0:N-1), w, wm

      pi = 4.0*atan(1.0)
      en = N

      if(N-1.eq.0) then
     xout(0) = xin(0)
      else
     alpha = 2.0*pi/en
     w = cmplx(cos(alpha),-sin(alpha))
     do 100 k=0, N-1
       wm = w**k
       xout(k) = xin(N-1)
       do 50 l=N-2, 0, -1
         xout(k) = xout(k)*wm + xin(l)
50         continue
100     continue
      endif

      return
      end
```

```
C                         SUBROUTINE: sample


C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE SAMPLES
C            OF A CONTINUOUS FUNCTION.  THE SAMPLES ARE RETURNED
C            TO THE MAIN PROGRAM IN THE ARRAY xin().


      subroutine sample(xin,N)
      complex xin(0:N-1)

      pi = 4.0*atan(1.0)
      en = N

C***********************************************************************

C  DEVELOP THE SAMPLING ALGORITHM IN THIS SPACE.  THE STATEMENTS
C  TYPED IN MUST FOLLOW STANDARD FORTRAN 77 RULES AND MAY USE
C  FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS: SIN(), COS(), ABS()...
C  AN EXAMPLE IS SHOWN BELOW.  THE INPUT SEQUENCE MUST BE STORED IN
C  THE ARRAY xin().  DFT.FOR MUST BE COMPILED AGAIN BEFORE EXECUTION
C  IF THIS SUBROUTINE IS USED.
C
C ***  EXAMPLE  ***
C
C      do 3 i=0, N-1
C        if(i.le.4) then
C          xin(i) = cmplx(i,0.0)
C        else
C          xin(i) = cmplx(0.0,0.0)
C        endif
C 3    continue

C***********************************************************************


      return
      end
```

```
C                             PRDGRM.FOR          VERSION: 2/03/88
C
C
C  PURPOSE:  THIS PROGRAM COMPUTES THE PERIODOGRAM OF A CAUSAL
C            SEQUENCE USING THE DISCRETE FOURIER TRANSFORM (DFT)
C            TECHNIQUE.  THE EQUATION FOR THE COMPUTATION OF THE
C            N-POINT PERIODOGRAM IS:  Sxx(k) = xk(k)*conjg(xk(k))/N
C            WHERE THE ARRAY xk() CONTAINS THE VALUES OF THE DFT
C            OF THE INPUT ARRAY xn(), I.E., xk() = DFT[xn()].  THE
C            PROGRAM CONSISTS OF A MAIN PROGRAM AND TWO SUBROUTINES.
C            THE SUBROUTINE dft COMPUTES THE DISCRETE FOURIER
C            TRANSFORM OF THE INPUT ARRAY, AND THE SUBROUTINE
C            sample ALLOWS THE USER TO GENERATE THE INPUT DATA BY
C            WRITING THE APPROPRIATE EQUATIONS. IF THE USER ELECTS
C            TO GENERATE THE INPUT DATA BY USING THE SUBROUTINE
C            sample, THE EQUATIONS MUST BE WRITTEN INTO THE
C            SUBROUTINE USING STANDARD FORTRAN 77 EXECUTABLE STATE-
C            MENTS AND THE INPUT DATA MUST BE STORED IN THE ARRAY
C            xn().  ALSO, IF EQUATIONS ARE WRITTEN INTO sample,
C            THE PROGRAM MUST BE COMPILED AGAIN BEFORE EXECUTION. THE
C            RESULTS OF THE PERIODIOGRAM COMPUTATION ARE STORED IN
C            THE ARRAY Sxx().  THE USER HAS THE OPTION OF CAUSING THE
C            OUTPUT TO BE CONVERTED TO DECIBELS. THE USER ALSO HAS
C            THE OPTION OF SELECTING ONE OF TWO OPERATING MODES: BATCH
C            OR TEST.  IN BATCH MODE THE AMOUNT OF INTERACTION WITH
C            THE USER IS MINIMIZED AND IT IS ASSUMED THAT THE INPUT
C            PARAMETERS HAVE BEEN STORED IN THE INPUT FILE 'PRDGRM.IN'.
C            IN THE TEST MODE THE USER IS PROMPTED FOR THE NAME
C            OF THE INPUT FILE OR HAS THE OPTION TO PERFORM A TRIAL
C            RUN USING THE DATA STORED IN THE FILE 'PRDGRM.TST'.
C            IT IS RECOMMENDED THAT FIRST-TIME USERS SELECT TEST
C            MODE WHEN PROMPTED, AND MAKE A TRIAL RUN WITH THE PRE-
C            STORED DATA.  ADDITIONALLY, THE TEST MODE ECHOES
C            PORTIONS OF THE INPUT DATA ONTO THE MONITOR TO ALLOW
C            VERIFICATION OF ITS ACCURACY.  THE OUTPUT IS STORED IN
C            TABULAR FORM IN THE FILE 'PRDGRM.OUT' AND IN A FORM
C            SUITABLE FOR PLOTTING IN THE FILE 'DFT.DAT'.
C
C
C*****************************  INPUT  ********************************
C
C
C  THIS PROGRAM ASSUMES THAT THERE ARE 'N' COMPLEX DATA POINTS IN THE
C  INPUT SEQUENCE.  THE INPUT SEQUENCE IS ASSUMED TO BE DEFINED IN
C  THE INTERVAL: 0 TO N - 1. IF THE INPUT SEQUENCE CONSISTS OF 'REAL'
C  NUMBERS, THE IMAGINARY PART IS STORED AS 0.0. THE VALUE 'N' AS WELL
C  AS THE OTHER PARAMETERS DESCRIBED BELOW SHOULD BE STORED IN THE
C  INPUT FILE 'PRDGRM.IN'.  ALL OF THE READ STATEMENTS USED BY THIS
C  PROGRAM REQUIRE FORMATTED INPUT.  PARTICULAR ATTENTION SHOULD BE
```

```
C  PAID TO THESE FORMATS, ESPECIALLY THE USE OF THE DECIMAL POINT TO
C  DISTINGUISH BETWEEN 'REAL' AND INTEGER DATA.
C
C
C  NAME        TYPE       RANGE (ARRAYS)       RESTRICTIONS
C  ----        ----       --------------     ----------------
C  N          INTEGER                          1 <= N <= 256
C  dsorce     CHARACTER                         'F'  OR  'S'
C  yscal      CHARACTER                        'STD' OR 'LOG'
C  xn()       COMPLEX       0,1,2...N-1        1 <= N <= 256
C
C  WHERE:
C
C  N = AN INTEGER THAT SPECIFIES THE NUMBER OF VALUES IN THE INPUT
C     SEQUENCE.
C
C  dsorce = A CHARACTER VALUE OF 'F' OR 'S' DENOTING WHETHER THE
C           INPUT DATA IS TO BE READ FROM A FILE (F) OR TO BE
C           GENERATED (S) BY A USER-DEFINED EQUATION LOCATED IN THE
C           SUBROUTINE sample.
C
C  yscal = A CHARACTER STRING SPECIFYING THE DESIRED MAGNITUDE OPTION:
C          'STD' WILL PRODUCE STANDARD MAGNITUDE OUTPUT;
C          'LOG' WILL PRODUCE MAGNITUDE EXPRESSED IN DECIBELS (dB).
C
C  xn()   =  THE ARRAY OF COMPLEX INPUT DATA.  IF dsorce = 'F'
C           IS SPECIFIED THE USER MUST SUPPLY THE N INPUT
C           VALUES IN THE FILE.  IF dsorce = 'S' THEN THE USER
C           HAS ELECTED TO GENERATE THE INPUT SEQUENCE BY
C           WRITING THE APPROPRIATE FORTRAN STATEMENTS IN THE
C           SPACE ALLOCATED IN SUBROUTINE sample.  IF THIS METHOD
C           OF DATA GENERATION IS ELECTED THE PROGRAM MUST BE
C           RECOMPILED BEFORE EXECUTION.
C
C  NOTE:  THE INPUT FORMAT STATEMENTS OCCUR IN THE MAIN PROGRAM
C         FOLLOWING THE CAPTION: ********  INPUT FORMAT  ********.
C         THE FORM OF THE INPUT DATA FILE IS:
C
C  LINE#               ENTRIES                  FORMAT
C  -----               -------                  ------
C    1              N,dsorce,yscal           i4,t11,a1,t21,a3
C  2...N+1              xn()                    2f10.0
C
C
C  NOTES 1.  LINES 2...N+1 ARE ONLY REQUIRED IF dsorce = 'F'.
C            IF dsorce = 'S' THEN THE USER HAS ELECTED TO GENERATE
C            THE N VALUES OF xn() IN SUBROUTINE sample.  THE USER
C            MUST PROVIDE THE APPROPRIATE FORTRAN STATEMENTS IN
C            SUBROUTINE sample AND THE VALUES MUST BE STORED IN THE
C            ARRAY xn().
C
```

```
C       2.  THE FORMAT f10.0 USED FOR INPUT DATA PERMITS THE
C           DECIMAL POINT TO BE PLACED ANYWHERE IN THE FIELD OF TEN
C           COLUMNS AND ALSO ALLOWS THE EXPONENTIAL FORMAT TO BE
C           USED (E.G., 3146.2 = 3.1462E+03).
C
C
C*****************************  OUTPUT   ******************************
C
C
C  THE INPUT DATA AS WELL AS THE OUTPUT DATA ARE STORED IN TABULAR
C  FORM IN THE FILE 'PRDGRM.OUT'.  ADDITIONALLY, THE INPUT SEQUENCE
C  (REAL AND IMAGINARY) AND THE OUTPUT SEQUENCE ARE STORED IN THE FILE
C  'PRDGRM.DAT' TO FACILITATE PLOTTING BY A SEPARATE, USER SUPPLIED
C  PROGRAM. THE FORMAT OF THE DATA IN 'PRDGRM.DAT' IS: e12.6,2x,e12.6.
C  THE FIRST ENTRY CORRESPONDS TO THE ORDINATE VALUE AND THE SECOND
C  ENTRY, THE ABSCISSA VALUE. ADDITIONAL HEADER INFORMATION IS WRITTEN
C  INTO 'PRDGRM.DAT' TO ALLOW FOR CONTROL AND LABELING OF EACH PLOT.
C
C
C*****************************  EXAMPLE  ******************************
C
C
C  THE INPUT PARAMETERS BELOW ARE STORED IN THE INPUT FILE
C  'PRDGRM.TST'.  THERE ARE EIGHT POINTS IN THE INPUT SEQUENCE AND
C  THE GOAL IS TO CALCULATE THE PERIODOGRAM OF THE DATA.
C
C
C  005       F         STD
C  0.0       0.0
C  1.0       0.0
C  2.0       0.0
C  3.0       0.0
C  4.0       0.0
C
C
C  THE RESULTING OUTPUT FILE 'PRDGRM.DAT' IS:
C
C  INPUT DATA SOURCEFILE: PRDGRM.TST
C  VALUE OF N =   5     dsorce = F     MAGNITUDE OPTION = STD
C
C
C                     INPUT DATA
C                        xn()
C
C      n      REAL          IMAGINARY
C      0     .0000E+00     .0000E+00
C      1     .1000E+01     .0000E+00
C      2     .2000E+01     .0000E+00
C      3     .3000E+01     .0000E+00
C      4     .4000E+01     .0000E+00
C
C
```

```
C
C
C                 OUTPUT DATA
C
C       k      Sxx(k)
C       0    .2000E+02
C       1    .3618E+01
C       2    .1382E+01
C       3    .1382E+01
C       4    .3618E+01
C
C
C  FOR ILLUSTRATIVE PURPOSES THE INPUT SEQUENCE xn() COULD HAVE
C  BEEN GENERATED BY SPECIFYING dsorce = 'S' AND WRITING THE
C  APPROPRIATE FORTRAN STATEMENTS INTO SUBROUTINE sample.  THE
C  STATEMENTS THAT COULD BE USED TO ACCOMPLISH THIS ARE WRITTEN
C  INTO THE SUBROUTINE BUT ARE 'COMMENTED OUT'.
C
C
C*************************  MAIN PROGRAM  ******************************


      character infile*12, ylabl*14, yscal*3, mode*1, dsorce*1
      character title*16
      complex xn(0:255), xk(0:255)
      real Sxx(0:255), nn, kk

C  PROMPT USER FOR MODE: BATCH OR TEST.

      write(*,1115)
      read(*,1117) mode
      if((mode.eq.'y').or.(mode.eq.'Y')) then
     mode = 'Y'
     write(*,1118)
     read(*,1119) infile
      else
     infile = 'PRDGRM.IN'
      endif

C  UNIT=1 DEFINED AS INPUT FILE.  UNITS=2,3 DEFINED AS OUTPUT FILES.

      open(unit=1,file=infile,status='old',iostat=ierr,err=999)
      open(unit=2,file='PRDGRM.OUT')
      open(unit=3,file='PRDGRM.DAT')

C  READ INPUT PARAMETERS AND CONDUCT ERROR CHECKS.

      read(1,1000) N, dsorce, yscal
```

```
       if((N.lt.1).or.(N.gt.256)) then
      write(*,1010) N
      stop 'The allowed values for N are:  1 <= N <= 256'
       endif

       if((dsorce.eq.'f').or.(dsorce.eq.'F')) then
      dsorce = 'F'
       elseif((dsorce.eq.'s').or.(dsorce.eq.'S')) then
      dsorce = 'S'
       else
      write(*,1009) dsorce
      stop 'The allowed values for dsorce are: ''S'' or ''F''.'
       endif

       if((yscal.eq.'std').or.(yscal.eq.'STD')) then
      title = '  Periodogram '
      yscal = 'STD'
      ylabl = '  Sxx(k)    '
       elseif((yscal.eq.'log').or.(yscal.eq.'LOG')) then
      title = ' Log Periodogram'
      yscal = 'LOG'
      ylabl = 'Sxx(k) (dB)'
       else
      write(*,1128) yscal
      stop 'Error, yscal must be the string ''STD'' or ''LOG''.'
       endif

C  FOR dsorce = 'F' READ THE INPUT SEQUENCE FROM THE INPUT FILE.
C  FOR dsorce = 'S' CALL sample TO GENERATE THE INPUT SEQUENCE.
C  THE INPUT SEQUENCE IS STORED IN THE ARRAY xn().

       if(dsorce.eq.'F') then
      read(1,1001) (xn(i),i=0,N-1)
       else
      call sample(xn,N)
       endif

C  DEFINE CONSTANTS

       k = 8
       en = N
       pi = 4.0*atan(1.0)
       numplts = 3

C  FOR TEST MODE ECHO INPUT DATA ONTO MONITOR (UNIT = *).

       if(mode.eq.'Y') then
      write(*,1016) infile
      if(N.lt.8) k = N
      write(*,1017) N, dsorce, yscal
      write(*,1012) k
      write(*,1015)
```

```
     do 4 i=0, k-1
       write(*,1020) i, xn(i)
4    continue
     endif

C  WRITE THE INPUT SEQUENCE INTO FILE: PRDGRM.DAT.

      write(3,2000) numplts
      write(3,2001) N
      write(3,*) 'INPUT SEQUENCE (REAL)'
      write(3,*) 'SAMPLE # (n)'
      write(3,*) 'REAL (xn)'
      do 55 i=0, N-1
        nn = i
        write(3,2010) nn, real(xn(i))
55    continue

      write(3,2001) N
      write(3,*) 'INPUT SEQUENCE (IMAGINARY)'
      write(3,*) 'SAMPLE # (n)'
      write(3,*) 'IMAGINARY (xn)'
      do 56 i=0, N-1
        nn = i
        write(3,2010) nn, aimag(xn(i))
56    continue

C  WRITE INPUT DATA INTO OUTPUT FILE: PRDGRM.OUT.

      write(2,1016) infile
      write(2,1017) N, dsorce, yscal
      write(2,1014)
      write(2,1015)
      do 59 i=0, N-1
        write(2,1020) i, xn(i)
59    continue

C  CALL dft TO COMPUTE THE DISCRETE FOURIER TRANSFORM OF THE
C  INPUT SEQUENCE.

      call dft(N,xn,xk)

C  THE PERIODOGRAM COMPUTATION RESULTS FROM THE EQUATION:
C  Sxx(k) = xk(k)*conjg(xk(k))/N.  THE SEQUENCE Sxx(k) IS
C  CONVERTED TO DECIBELS IF yscal = 'LOG'.

      do 60 k=0, N-1
        Sxx(k) = xk(k)*conjg(xk(k))/en
     if(yscal.eq.'LOG') then
       if(Sxx(k).gt.1.0e-10) then
         Sxx(k) = 10.0*log10(Sxx(k))
       else
         Sxx(k) = -100.0
       endif
     endif
60    continue

C  WRITE RESULTS INTO OUTPUT FILE: PRDGRM.DAT.

      write(3,2001) N
      write(3,2002) title
      write(3,*) ' k '
      write(3,*) ylabl
      do 57 k=0, N-1
     kk = k
     write(3,2010) kk, Sxx(k)
57    continue

C  WRITE RESULTS INTO OUTPUT FILE: PRDGRM.OUT.

      write(2,1025) ylabl
      do 5 k=0, N-1
     write(2,1030) k, Sxx(k)
5     continue

      write(*,1019)
999   close(unit=1)
      close(unit=2)
      close(unit=3)

      if(ierr.gt.0) then
        write(*,1116) infile, ierr
      endif

C  ********  INPUT FORMAT  ********

1000  format(i3,t11,a1,t21,a3)
1001  format(2f10.0)

C  ********************************

1009  format(1x,'dsorce = ',a1,2x,'Error, illegal value for dsorce.')
1010  format(1x,'N = ',i3)
1012 0format(/,' THE FIRST ',i1,' VALUES OF xn() ARE LISTED BELOW.',
     1/,' VERIFY THAT THE DATA IS CORRECT.',/)
1014  format(//,t22,'INPUT DATA',/,t25,'xn()',//)
1015  format(t8,'n',t15,'REAL',t28,'IMAGINARY')
1016  format(//////,' INPUT DATA SOURCEFILE: ',a12)
1017 0format(' VALUE OF N = ',i3,5x,'dsorce = ',a1,5x,'MAGNITUDE ',
     1'OPTION = ',a3)
```

```
1019 0format(//,' TABULAR OUTPUT DATA IS STORED IN FILE: PRDGRM.OUT.',
     1/,' PLOTTING DATA IS STORED IN FILE: PRDGRM.DAT')
1020  format(4x,i4,2(4x,e10.4))
1025  format(///,t19,'OUTPUT DATA',//,t7,'k',t12,a11)
1030  format(t5,i3,4x,e10.4)
1115 0format(1x,'DO YOU WISH TO RUN THIS PROGRAM IN TEST',
     1' MODE ?    (Y/N) <CR> : ',\,)
1116 0format(///,1x,'ERROR OPENING INPUT FILE: ',a12,/,1x,'PROGRAM',
     1' TERMINATED.',//,1x,'ERROR CODE:',i4,/////)
1117  format(a1)
1118 0format(/////,1x,'TYPE THE NAME OF YOUR DATA FILE FOLLOWED',
     1' BY <CR>.',/,' IF YOU DESIRE TO MAKE A TEST RUN USING THE',
     2' SAMPLE DATA ALREADY STORED',/,' IN THE FILE: PRDGRM.TST',
     3'          TYPE: PRDGRM.TST <CR>',/,' FILENAME: ',\,)
1119  format(a12)
1128  format(//////,t2,'The value of yscal is: ',a3,'.')
2000  format(i1)
2001  format(i3)
2002  format(a16)
2003  format(a8)
2010  format(e12.6,2x,e12.6)

      end


C                          SUBROUTINE:  dft


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY
C            xn(), COMPUTES THE DISCRETE FOURIER TRANSFORM (DFT)
C            OF THE ARRAY, AND RETURNS THE RESULTING SEQUENCE IN THE
C            COMPLEX ARRAY xk().


      subroutine dft(N,xn,xk)
      complex xn(0:N-1), xk(0:N-1), w, wm

      pi = 4.0*atan(1.0)
      en = N

      if(N-1.eq.0) then
     xk(0) = xn(0)
      else
     alpha = 2.0*pi/en
     w = cmplx(cos(alpha),-sin(alpha))
```

```
    do 100 k=0, N-1
      wm = w**k
      xk(k) = xn(N-1)
         do 50 l=N-2, 0, -1
        xk(k) = xk(k)*wm + xn(l)
50       continue
100     continue
      endif

      return
      end


C                         SUBROUTINE:  sample
C
C
C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE SAMPLES
C            OF A CONTINUOUS FUNCTION.  THE SAMPLES ARE RETURNED
C            TO THE MAIN PROGRAM IN THE ARRAY xn().


      subroutine sample(xn,N)
      complex xn(0:N-1)

      pi = 4.0*atan(1.0)
      en = N

C************************************************************************

C  DEVELOP THE SAMPLING ALGORITHM IN THIS SPACE.  THE STATEMENTS
C  TYPED IN MUST FOLLOW STANDARD FORTRAN 77 RULES AND MAY USE
C  FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS: SIN(), COS(), ABS()...
C  AN EXAMPLE IS SHOWN BELOW.  THE INPUT DATA MUST BE STORED IN THE
C  ARRAY xn().  IF THIS SUBROUTINE IS USED, 'PRDGRM.FOR' MUST BE
C  COMPILED AGAIN BEFORE EXECUTION.
C
C ***  EXAMPLE  ***
C
C     do 3 i=0, N-1
C       xn(i) = cmplx(i,0.0)
C 3   continue

C************************************************************************


      return
      end
```

```
C                         CONCORDT.FOR          VERSION: 2/03/88
C
C
C  PURPOSE:  THIS PROGRAM PERFORMS ANY ONE OF THE FOLLOWING FOUR
C            COMPUTATIONS GIVEN TWO COMPLEX ARRAYS OF INPUT DATA:
C            LINEAR CONVOLUTION (LCON); LINEAR CORRELATION (LCOR);
C            CIRCULAR CONVOLUTION (CCON); OR CIRCULAR CORRELATION
C            (CCOR). CONVOLUTION IS PERFORMED BY COMPUTING THE DFT
C            OF EACH ARRAY, MULTIPLYING THE DFTs TOGETHER AND THEN
C            TAKING THE INVERSE DFT OF THE RESULTING ARRAY.
C            CORRELATION IS PERFORMED IN THE SAME MANNER EXCEPT
C            THAT THE CONJUGATE OF THE DFT OF ARRAY #1 IS MULTIPLIED
C            BY THE DFT OF ARRAY #2.  THE RESULT OF THIS COMPUTATION,
C            GIVEN THE INPUT SEQUENCES x1 AND x2, IS THE CORRELATION
C            SEQUENCE Rx1x2.  THE PROGRAM CONSISTS OF A MAIN
C            PROGRAM AND FIVE SUBROUTINES. THE SUBROUTINE zeropad
C            EXTENDS THE INPUT ARRAY PASSED TO IT BY ADDING AN
C            APPROPRIATE NUMBER OF ZEROES TO THE ORIGINAL INPUT DATA
C            TO CREATE AN ARRAY OF SUITABLE LENGTH FOR THE LINEAR
C            CONVOLUTION/CORRELATION ALGORITHMS.  THE SUBROUTINE
C            dft COMPUTES THE DISCRETE FOURIER TRANSFORM OF AN
C            ARRAY. THE SUBROUTINE invdft COMPUTES THE INVERSE
C            DISCRETE FOURIER TRANSFORM OF AN ARRAY.  THE TWO
C            SUBROUTINES sampl1 AND sampl2 ALLOW THE USER TO
C            GENERATE EITHER OF THE INPUT ARRAYS BY WRITING THE
C            APPROPRIATE EQUATIONS.  IF THE USER CHOOSES TO
C            GENERATE THE INPUT DATA BY USING EITHER OF THE sampl
C            SUBROUTINE(S), THE EQUATIONS MUST BE WRITTEN INTO THE
C            SUBROUTINE(S) USING STANDARD FORTRAN 77 EXECUTABLE
C            STATEMENTS AND THE VALUES GENERATED MUST BE STORED
C            IN THE ARRAYS xn1() AND xn2().  THE USER HAS THE
C            OPTION OF SELECTING ONE OF TWO OPERATING MODES: BATCH
C            OR TEST.  IN BATCH MODE THE AMOUNT OF INTERACTION
C            WITH THE USER IS MINIMIZED AND IT IS ASSUMED THAT THE
C            INPUT PARAMETERS HAVE BEEN STORED IN THE INPUT FILE
C            'CONCORDT.IN'.  IN TEST MODE THE USER IS PROMPTED
C            FOR THE NAME OF THE INPUT FILE AND HAS THE OPTION
C            TO PERFORM A TRIAL RUN USING THE DATA STORED IN
C            THE FILE 'CONCORDT.TST'.  IT IS RECOMMENDED THAT FIRST-
C            TIME USERS SELECT THE TEST MODE AND PERFORM A TRIAL
C            RUN WITH THE PRESTORED DATA.  THE TEST MODE ECHOES
C            PORTIONS OF THE INPUT DATA ONTO THE MONITOR TO ALLOW
C            VERIFICATION OF ITS ACCURACY.  THE OUTPUT OF THE
C            PROGRAM 'CONCORDT.FOR' IS STORED IN THE ARRAY xn3().
C            THE OUTPUT IS STORED IN TABULAR FORM IN THE FILE
C            'CONCORDT.OUT' AND IN A FORM SUITABLE FOR PLOTTING
C            IN THE FILE 'CONCORDT.DAT'.
C
C
C******************************   INPUT   **********************************

C
```

```
C
C THIS PROGRAM ASSUMES THAT THERE ARE TWO SEQUENCES OF INPUT DATA
C STORED IN THE ARRAYS xn1() AND xn2() OF LENGTH 'N1' AND 'N2',
C RESPECTIVELY.  THE ARRAYS ARE ASSUMED TO BE COMPLEX.  IF THE
C ARRAYS CONTAIN 'REAL' VALUES ONLY, THEN THE IMAGINARY PART IS
C STORED AS 0.0. THE INPUT SEQUENCES ARE ASSUMED TO BE DEFINED
C IN THE INTERVALS 0 TO N1-1 AND 0 TO N2-1, RESPECTIVELY.
C THIS PROGRAM ALLOWS THE USER THE OPTION OF EITHER READING THE
C THE INPUT ARRAYS FROM A DATA FILE OR OF GENERATING THE INPUT
C VALUES FROM AN ITERATIVE EQUATION IN THE sampl SUBROUTINE(S).
C THE PARAMETERS DESCRIBED BELOW ALLOW THE USER TO SELECT THE
C DESIRED OPTIONS AND THESE PARAMETERS MUST BE STORED IN THE INPUT
C FILE 'CONCORDT.IN'.  ALL OF THE READ STATEMENTS USED BY THIS
C PROGRAM REQUIRE FORMATTED INPUT.  PARTICULAR ATTENTION SHOULD BE
C PAID TO THESE FORMATS, ESPECIALLY THE USE OF THE DECIMAL POINT TO
C DISTINGUISH BETWEEN 'REAL' AND INTEGER DATA.
C
C
C NAME         TYPE       RANGE (ARRAYS)         RESTRICTIONS
C ----         ----       --------------       ----------------
C N1          INTEGER                            1 <= N1 <= 128
C dsrce1      CHARACTER                           'F'  OR  'S'
C N2          INTEGER                            1 <= N2 <= 128
C dsrce2      CHARACTER                           'F'  OR  'S'
C option      CHARACTER                      ONE OF THE FOLLOWING:
C                                        'LCON' 'LCOR' 'CCON' 'CCOR'
C
C xn1()       COMPLEX      0,1,...,N1-1          1 <= N1 <= 128
C xn2()       COMPLEX      0,1,...,N2-1          1 <= N2 <= 128
C
C WHERE:
C
C N1 = AN INTEGER THAT SPECIFIES THE NUMBER OF POINTS OF INPUT
C      DATA TO BE STORED IN THE ARRAY xn1().
C
C dsrce1 =  A CHARACTER VALUE OF 'F' OR 'S' DENOTING WHETHER THE
C           INPUT ARRAY xn1() IS TO BE READ FROM A FILE (F) OR TO
C           BE GENERATED (S) BY A USER-DEFINED EQUATION LOCATED IN
C           THE SUBROUTINE sampl1.
C
C N2 = AN INTEGER THAT SPECIFIES THE NUMBER OF POINTS OF INPUT
C      DATA TO BE STORED IN THE ARRAY xn2().
C
C dsrce2 =  A CHARACTER VALUE OF 'F' OR 'S' DENOTING WHETHER THE
C           INPUT ARRAY xn2() IS TO BE READ FROM A FILE (F) OR TO
C           BE GENERATED (S) BY A USER-DEFINED EQUATION LOCATED IN
C           THE SUBROUTINE sampl2.
C
```

```
C  option = A CHARACTER STRING OF FOUR LETTERS DENOTING THE
C           COMPUTATION DESIRED.  'LCON' = LINEAR CONVOLUTION
C                                 'LCOR' = LINEAR CORRELATION
C                                 'CCON' = CIRCULAR CONVOLUTION
C                                 'CCOR' = CIRCULAR CORRELATION
C
C  xn1() =  THE FIRST ARRAY OF COMPLEX INPUT DATA. IF dsrce1 = 'F'
C           IS SPECIFIED THE USER MUST SUPPLY THE N1 INPUT
C           VALUES IN THE FILE.  IF dsrce1 = 'S' THE USER HAS
C           ELECTED TO GENERATE THE INPUT DATA BY PROVIDING
C           THE APPROPRIATE FORTRAN STATEMENTS IN THE SPACE
C           ALLOCATED IN SUBROUTINE sampl1.  IF THIS METHOD
C           OF DATA GENERATION IS ELECTED THE PROGRAM MUST BE
C           RECOMPILED BEFORE EXECUTION.
C
C  xn2() =  THE SECOND ARRAY OF COMPLEX INPUT DATA.  IF dsrce2 =
C           'S' IS SPECIFIED THE USER HAS ELECTED TO PROVIDE THE
C           APPROPRIATE FORTRAN STATEMENTS IN THE SPACE ALLOCATED
C           IN SUBROUTINE sampl2.  IF THIS METHOD OF DATA
C           GENERATION IS ELECTED THE PROGRAM MUST BE RECOMPILED
C           BEFORE EXECUTION.  IF dsrce2 = 'F' THEN THE USER MUST
C           SUPPLY THE N2 INPUT VALUES IN THE FILE.
C
C  NOTE:  THE INPUT FORMAT STATEMENTS OCCUR IN THE MAIN PROGRAM
C         FOLLOWING THE CAPTION: ********  INPUT FORMAT  ********.
C         THE FORM OF THE INPUT DATA FILE IS:
C
C  LINE#             ENTRIES                      FORMAT
C  -----             -------                      ------
C    1              N1,dsrce1                   i3,t11,a1
C    2          N2,dsrce2,option            i3,t11,a1,t21,a4
C  NOTE 1            xn1()                        2f10.0
C  NOTE 2            xn2()                        2f10.0
C
C  NOTES 1.  IF dsrce1 = 'F' THEN THE LINES 3...N1+2 MUST CONTAIN
C            THE VALUES TO BE READ INTO THE ARRAY xn1(). EACH VALUE
C            IS READ AS A COMPLEX NUMBER, I.E.,  REAL   IMAGINARY.
C            IF dsrce1 = 'S' THEN THE USER HAS ELECTED TO GENERATE
C            THE VALUES FOR xn1() IN THE SUBROUTINE sampl1. THE USER
C            MUST THEN PROVIDE THE APPROPRIATE FORTRAN STATEMENTS
C            IN SUBROUTINE sampl1 TO GENERATE xn1().
C
C        2.  IF dsrce2 = 'F' THEN THE NEXT N2 LINES CONTAIN THE
C            VALUES TO BE READ INTO THE ARRAY xn2().  EACH VALUE IS
C            READ AS A COMPLEX NUMBER, I.E.,  REAL   IMAGINARY.  IF
C            dsrce2 = 'S' THEN THE USER HAS ELECTED TO GENERATE THE
C            VALUES FOR xn2() IN THE SUBROUTINE sampl2. THE USER
C            MUST THEN PROVIDE THE APPROPRIATE FORTRAN STATEMENTS
C            IN SUBROUTINE sampl2 TO GENERATE THE ARRAY xn2().
C
```

```
C       3.  THE FORMAT 2f10.0 USED FOR INPUT DATA PERMITS THE
C           DECIMAL POINT TO BE PLACED ANYWHERE IN THE FIELD OF TEN
C           COLUMNS AND ALSO ALLOWS THE EXPONENTIAL FORMAT TO BE
C           USED (E.G., 3146.2 = 3.1462E+03).
C
C       4.  IF option = 'CCON' OR 'CCOR' N1 MUST BE EQUAL TO N2.
C
C
C******************************  OUTPUT   ********************************

C
C
C  THE INPUT DATA AS WELL AS THE OUTPUT DATA ARE STORED IN TABULAR
C  FORM IN THE FILE 'CONCORDT.OUT'. ADDITIONALLY, THE INPUT SEQUENCES
C  AND THE OUTPUT SEQUENCE ARE WRITTEN INTO THE FILE 'CONCORDT.DAT'
C  TO FACILITATE PLOTTING BY A SEPARATE, USER SUPPLIED PROGRAM.  THE
C  FORMAT OF THE DATA IN 'CONCORDT.DAT' IS:  e12.6, 2x, e12.6.  THE
C  FIRST ENTRY CORRESPONDS TO THE ORDINATE VALUE AND THE SECOND ENTRY,
C  THE ABSCISSA VALUE.  ADDITIONAL HEADER INFORMATION IS WRITTEN INTO
C  'CONCORDT.DAT' TO ALLOW FOR CONTROL AND LABELING OF EACH PLOT.
C
C
C******************************  EXAMPLE  ********************************

C
C
C  THE INPUT PARAMETERS BELOW ARE STORED IN THE INPUT FILE
C  'CONCORDT.TST'.  THE PROGRAM READS THE FIRST 4 VALUES INTO xn1()
C  (dsrce1 = 'F', N1 = 4), AND READS THE NEXT 5 VALUES INTO xn2()
C  (dsrce2 = 'F', N2 = 5).  THE GOAL IS TO CALCULATE THE LINEAR
C  CONVOLUTION OF THE TWO INPUT ARRAYS.
C
C
C  004       F
C  005       F         LCON
C  1.0       0.0
C  2.0       0.0
C  3.0       0.0
C  4.0       0.0
C  5.0       0.0
C  4.0       0.0
C  3.0       0.0
C  2.0       0.0
C  1.0       0.0
C
C
C  THE RESULTING OUTPUT DATA FILE 'CONCORDT.OUT' IS:
C
C  INPUT DATA SOURCEFILE: CONCORDT.TST
C  N1 =   4     dsrce1 = F
C  N2 =   5     dsrce2 = F
C  option = LCON
```

```
C
C
C                 INPUT DATA
C
C                    xn1()
C   n      REAL              IMAGINARY
C   0     .100000E+01     .000000E+00
C   1     .200000E+01     .000000E+00
C   2     .300000E+01     .000000E+00
C   3     .400000E+01     .000000E+00
C
C                    xn2()
C   n      REAL              IMAGINARY
C   0     .500000E+01     .000000E+00
C   1     .400000E+01     .000000E+00
C   2     .300000E+01     .000000E+00
C   3     .200000E+01     .000000E+00
C   4     .100000E+01     .000000E+00
C
C
C                 OUTPUT DATA
C
C                    xn3()
C   n      REAL              IMAGINARY
C   0     .500000E+01     .953674E-06
C   1     .140000E+02    -.303457E-05
C   2     .260000E+02    -.756009E-05
C   3     .400000E+02    -.404610E-05
C   4     .300000E+02     .217716E-05
C   5     .200000E+02     .762858E-05
C   6     .110000E+02     .892130E-05
C   7     .400001E+01     .472045E-05
C
C
C  FOR ILLUSTRATIVE PURPOSES THE INPUT SEQUENCES xn1() AND xn2()
C  COULD HAVE BEEN GENERATED BY SPECIFYING dsrce# = 'S' AND WRITING
C  THE APPROPRIATE FORTRAN STATEMENTS INTO THE sampl# SUBROUTINES.
C  THE STATEMENTS THAT COULD BE USED TO ACCOMPLISH THIS ARE WRITTEN
C  INTO THE RESPECTIVE SUBROUTINES BUT ARE 'COMMENTED OUT'.
C
C
C***************************  MAIN PROGRAM  ***************************


      character infile*12, option*4, mode*1, dsrce1*1, dsrce2*1
      character title*20
      complex xn1(0:255), xn2(0:255), xn3(0:255)
      complex xk1(0:255), xk2(0:255), xk3(0:255)
      real nn
```

```
C  PROMPT USER FOR MODE: BATCH OR TEST.

      write(*,1115)
      read(*,1117) mode
      if((mode.eq.'y').or.(mode.eq.'Y')) then
     mode = 'Y'
     write(*,1118)
     read(*,1119) infile
      else
     infile = 'CONCORDT.IN'
      endif

C  UNIT=1 DEFINED AS INPUT FILE.  UNITS=2,3 DEFINED AS OUTPUT FILES.

      open(unit=1,file=infile,status='old',iostat=ierr,err=999)
      open(unit=2,file='CONCORDT.OUT')
      open(unit=3,file='CONCORDT.DAT')

C  READ INPUT PARAMETERS AND CONDUCT ERROR CHECKS.

      read(1,1000) N1, dsrce1
      read(1,1001) N2, dsrce2, option

      if((dsrce1.eq.'f').or.(dsrce1.eq.'F')) then
     dsrce1 = 'F'
      elseif((dsrce1.eq.'s').or.(dsrce1.eq.'S')) then
     dsrce1 = 'S'
      else
     write(*,1009) 'dsrce1 = ', dsrce1
     stop 'The allowed values for dsrce1 are: ''F'' or ''S''.'
      endif

      if((dsrce2.eq.'f').or.(dsrce2.eq.'F')) then
     dsrce2 = 'F'
      elseif((dsrce2.eq.'s').or.(dsrce2.eq.'S')) then
     dsrce2 = 'S'
      else
     write(*,1009) 'dsrce2 = ', dsrce2
     stop 'The allowed values for dsrce2 are: ''F'' or ''S''.'
      endif

      if((option.eq.'ccon').or.(option.eq.'CCON')) then
     option = 'CCON'
     title = 'Circular Convolution'
      elseif((option.eq.'ccor').or.(option.eq.'CCOR')) then
     option = 'CCOR'
     title = 'Circular Correlation'
      elseif((option.eq.'lcon').or.(option.eq.'LCON')) then
     option = 'LCON'
     title = 'Linear Convolution'
      elseif((option.eq.'lcor').or.(option.eq.'LCOR')) then
     option = 'LCOR'
```

```
      title = 'Linear Correlation'
       else
      write(*,1011) option
      stop 'The allowed values for option are: CCON,CCOR,LCON,LCOR.'
       endif

       if((N1.lt.1).or.(N1.gt.128)) then
      write(*,1010) 'N1 = ', N1
      stop 'The allowed values for N1 are:  1 <= N1 <= 128'
       endif

       if((N2.lt.1).or.(N2.gt.128)) then
      write(*,1010) 'N2 = ', N2
      stop 'The allowed values for N2 are:  1 <= N2 <= 128'
       endif

       if((option.eq.'CCON').or.(option.eq.'CCOR')) then
      if(N1.ne.N2) then
        write(*,1008) option, N1, N2
        stop 'For option = ''CCOR'' or ''CCON'' N1 must equal N2.'
      endif
      N3 = N1
       endif

C  DEFINE CONSTANTS.

       k = 8
       numplts = 6

C  FOR dsrce# = 'F' READ INPUT SEQUENCE(S) FROM THE DATA FILE.
C  FOR dsrce# = 'S' CALL sampl# TO GENERATE THE INPUT SEQUENCE(S).
C  THE INPUT SEQUENCES ARE STORED IN THE ARRAYS xn1(), xn2().

       if(dsrce1.eq.'F') then
      read(1,1002) (xn1(i),i=0,N1-1)
       else
      call sampl1(xn1,N1)
       endif

       if(dsrce2.eq.'F') then
      read(1,1002) (xn2(i),i=0,N2-1)
       else
      call sampl2(xn2,N2)
       endif


C  FOR TEST MODE ECHO INPUT DATA ONTO MONITOR (UNIT = *).

       if(mode.eq.'Y') then
      write(*,1016) infile
      if((N1.lt.8).or.(N2.lt.8)) k = min(N1,N2)
      write(*,1017) 'N1 = ', N1, 'dsrce1 = ', dsrce1
      write(*,1017) 'N2 = ', N2, 'dsrce2 = ', dsrce2
      write(*,1018) option
      write(*,1012) k
      write(*,1013)
      do 3 i=0, k-1
        write(*,1020) i, xn1(i), xn2(i)
3         continue
       endif

C  WRITE THE INPUT SEQUENCES INTO FILE: CONCORDT.DAT.

       write(3,2000) numplts
       write(3,2001) N1
       write(3,*) 'INPUT SEQUENCE xn1 (REAL)'
       write(3,*) 'SAMPLE # (n)'
       write(3,*) 'REAL xn1()'
       do 54 i=0, N1-1
      nn = i
      write(3,2010) nn, real(xn1(i))
54     continue

       write(3,2001) N1
       write(3,*) 'INPUT SEQUENCE xn1 (IMAGINARY)'
       write(3,*) 'SAMPLE # (n)'
       write(3,*) 'IMAG xn1()'
       do 55 i=0, N1-1
      nn = i
      write(3,2010) nn, aimag(xn1(i))
55     continue

       write(3,2001) N2
       write(3,*) 'INPUT SEQUENCE xn2 (REAL)'
       write(3,*) 'SAMPLE # (n)'
       write(3,*) 'REAL xn2()'
       do 56 i=0, N2-1
      nn = i
      write(3,2010) nn, real(xn2(i))
56     continue

       write(3,2001) N2
       write(3,*) 'INPUT SEQUENCE xn2 (IMAGINARY)'
       write(3,*) 'SAMPLE # (n)'
       write(3,*) 'IMAG xn2()'
       do 57 i=0, N2-1
      nn = i
      write(3,2010) nn, aimag(xn2(i))
57     continue

C  WRITE INPUT DATA INTO FILE: CONCORDT.OUT.

       write(2,1016) infile
       write(2,1017) 'N1 = ', N1, 'dsrce1 = ', dsrce1
      write(2,1017) 'N2 = ', N2, 'dsrce2 = ', dsrce2
      write(2,1018) option
      write(2,1014) 'INPUT DATA'
      write(2,1015) 'xn1()'
      do 60 i=0, N1-1
     write(2,1026) i, xn1(i)
60    continue
      write(2,1015) 'xn2()'
      do 61 i=0, N2-1
     write(2,1026) i, xn2(i)
61    continue

C   FOR LINEAR CONVOLUTION OR LINEAR CORRELATION BOTH INPUT ARRAYS
C   ARE ZERO-PADDED TO LENGTH N3 = N1 + N2 - 1.

      if((option.eq.'LCON').or.(option.eq.'LCOR')) then
     N3 = N1 + N2 - 1
     call zeropad(xn1,N1,N3)
     call zeropad(xn2,N2,N3)
      endif

C   COMPUTE THE DFT OF BOTH INPUT SEQUENCES.

     call dft(N3,xn1,xk1)
     call dft(N3,xn2,xk2)

C   PERFORM CONVOLUTION COMPUTATION.

      if((option.eq.'LCON').or.(option.eq.'CCON')) then
     do 22 i=0, N3-1
       xk3(i) = xk1(i)*xk2(i)
22      continue
     call invdft(N3,xk3,xn3)
      endif

C   PERFORM CORRELATION COMPUTATION.

      if((option.eq.'LCOR').or.(option.eq.'CCOR')) then
     do 23 i=0, N3-1
       xk1(i) = conjg(xk1(i))
       xk3(i) = xk1(i)*xk2(i)
23      continue
     call invdft(N3,xk3,xn3)
      endif

C   WRITE RESULTS INTO FILE: CONCORDT.DAT.

      write(3,2001) N3
      write(3,2003) title
      write(3,*) 'SAMPLE # (n)'
      write(3,*) 'REAL xn3()'
      do 58 i=0, N3-1
```

```
    nn = i
    write(3,2010) nn, real(xn3(i))
58     continue

       write(3,2001) N3
       write(3,2003) title
       write(3,*) 'SAMPLE # (n)'
       write(3,*) 'IMAG xn3()'
       do 59 i=0, N3-1
    nn = i
    write(3,2010) nn, aimag(xn3(i))
59     continue

C  WRITE RESULTS INTO FILE: CONCORDT.OUT.

       write(2,1014) 'OUTPUT DATA'
       write(2,1015) 'xn3()'
       do 62 i=0, N3-1
    write(2,1026) i, xn3(i)
62     continue

       write(*,1019)
999    close(unit=1)
       close(unit=2)
       close(unit=3)
       if(ierr.gt.0) then
    write(*,1116) infile, ierr
       endif

C  ********  INPUT FORMAT  ********

1000  format(i3,t11,a1)
1001  format(i3,t11,a1,t21,a4)
1002  format(2f10.0)

C  ********************************

1008 0format(' option = ',a4,',  N1 = ',i4,',  N2 = ',i4,',   Error,',
     1' N1 is not equal to N2.')
1009  format(1x,a9,a1,'  Error, value not allowed.')
1010  format(1x,a5,i4,2x,'Error, value not allowed.')
1011  format(1x,'option = ',a4,2x,'Error, illegal value for option.')
1012 0format(/,t2,'THE FIRST',i3,' VALUES OF INPUT DATA ARE LISTED ',
     1/,' BELOW,  VERIFY THAT THE DATA IS CORRECT.',/)
1013 0format(t21,'xn1()',t53,'xn2()',/,t4,'n',t11,'REAL',t27,
     1'IMAGINARY',t43,'REAL',t59,'IMAGINARY')
1014  format(//,t20,a11,/)
1015  format(/,t21,a7,/,t6,'n',t13,'REAL',t29,'IMAGINARY')
1016  format(///,' INPUT DATA SOURCEFILE: ',a12)
1017  format(t2,a5,i3,5x,a9,a1)
1018  format(t2,'option = ',a4)
```

```
1019 0format(//,' TABULAR OUTPUT DATA IS STORED IN FILE: CONCORDT.OUT.',
     1/,' PLOTTING DATA IS STORED IN FILE: CONCORDT.DAT.')
1020  format(t4,i1,4(4x,e12.6))
1026  format(t4,i3,2(4x,e12.6))
1115 0format(1x,'DO YOU WISH TO RUN THIS PROGRAM IN TEST',
     1' MODE ?    (Y/N) <CR> : ',\,)
1116 0format(///,1x,'ERROR OPENING INPUT FILE: ',a12,/,1x,'PROGRAM',
     1' TERMINATED.',//,1x,'ERROR CODE:',i4,/////)
1117  format(a1)
1118 0format(/////,1x,'TYPE THE NAME OF YOUR DATA FILE FOLLOWED',
     1' BY <CR>.',/,' IF YOU DESIRE TO MAKE A TEST RUN USING THE',
     2' SAMPLE DATA ALREADY STORED',/,' IN THE FILE: CONCORDT.TST',
     3'         TYPE: CONCORDT.TST <CR>',/,' FILENAME: ',\,)
1119  format(a12)
2000  format(i1)
2001  format(i3)
2003  format(a20)
2010  format(e12.6,2x,e12.6)

      end


C                         SUBROUTINE: zeropad


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY xn()
C            OF LENGTH N AND ZERO PADS THE ARRAY TO LENGTH N3.


      subroutine zeropad(xn,N,N3)
      complex xn(0:N3-1)

      do 33 i=N, N3-1
     xn(i) = cmplx(0.0,0.0)
33    continue

      return
      end


C                         SUBROUTINE: invdft


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY
C            xin() AND COMPUTES THE INVERSE DFT OF THE ARRAY.  THE
C            OUTPUT IS STORED IN THE COMPLEX ARRAY xout().


      subroutine invdft(N,xin,xout)
      complex xin(0:N-1), xout(0:N-1)

      en = N
```

```
C  COMPUTE THE COMPLEX CONJUGATE OF THE INPUT DATA.

      do 70 i=0, N-1
     xin(i) = conjg(xin(i))
70    continue

C  COMPUTE THE DISCRETE FOURIER TRANSFORM OF THE ARRAY.

      call dft(N,xin,xout)

C  COMPUTE THE COMPLEX CONJUGATE OF THE RESULTING ARRAY.

      do 80 i=0, N-1
     xout(i) = conjg(xout(i))/en
80    continue

      return
      end


C                    SUBROUTINE: dft


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY
C            xin(), COMPUTES THE DISCRETE FOURIER TRANSFORM (DFT)
C            OF THE ARRAY, AND RETURNS THE RESULTING SEQUENCE IN THE
C            COMPLEX ARRAY xout().


      subroutine dft(N,xin,xout)
      complex xin(0:N-1), xout(0:N-1), w, wm

      pi = 4.0*atan(1.0)
      en = N

      if(N-1.eq.0) then
     xout(0) = xin(0)
      else
     alpha = 2.0*pi/en
     w = cmplx(cos(alpha),-sin(alpha))
     do 100 k=0, N-1
       wm = w**k
       xout(k) = xin(N-1)
       do 50 l=N-2, 0, -1
         xout(k) = xout(k)*wm+xin(l)
50        continue
100     continue
      endif

      return
      end
```

```
C                          SUBROUTINE: sampl1

C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE SAMPLES
C            OF A CONTINUOUS FUNCTION AND STORE THEM IN THE ARRAY
C            xn1(). IF dsrce1 = 'S' THEN THE MAIN PROGRAM WILL CALL
C            THIS SUBROUTINE TO GENERATE THE VALUES FOR xn1().
C            IF dsrce1 DOES NOT EQUAL 'S' THEN THIS SUBROUTINE WILL
C            NOT BE CALLED BY THE MAIN PROGRAM.


      subroutine sampl1(xn1,N1)
      complex xn1(0:N1-1)

      pi = 4.0*atan(1.0)
      en1 = N1

C**********************************************************************

C  DEVELOP THE SAMPLING ALGORITHM FOR xn1() IN THIS SPACE. THE
C  STATEMENTS TYPED IN MUST FOLLOW STANDARD FORTRAN 77 RULES AND
C  MAY USE FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS: SIN(),COS()...
C  AN EXAMPLE OF AN ALGORITHM GENERATING VALUES FOR xn1() IS SHOWN.
C
C ***  EXAMPLE  ***
C
C     do 6 i=0, N1-1
C       xn1(i) = cmplx(i+1.0,0.0)
C 6   continue

C**********************************************************************


      return
      end


C                          SUBROUTINE: sampl2

C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE SAMPLES
C            OF A CONTINUOUS FUNCTION AND STORE THEM IN THE ARRAY
C            xn2(). IF dsrce2 = 'S' THEN THE MAIN PROGRAM WILL CALL
C            THIS SUBROUTINE TO GENERATE THE VALUES FOR xn2().
C            IF dsrce2 DOES NOT EQUAL 'S' THEN THIS SUBROUTINE WILL
C            NOT BE CALLED BY THE MAIN PROGRAM.
```

```
      subroutine sampl2(xn2,N2)
      complex xn2(0:N2-1)

      pi = 4.0*atan(1.0)
      en2 = N2

C*************************************************************************

C  DEVELOP THE SAMPLING ALGORITHM FOR xn2() IN THIS SPACE. THE
C  STATEMENTS TYPED IN MUST FOLLOW STANDARD FORTRAN 77 RULES AND
C  MAY USE FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS: SIN(),COS()...
C  AN EXAMPLE OF AN ALGORITHM GENERATING VALUES FOR xn2() IS SHOWN.
C
C ***  EXAMPLE  ***
C
C     do 7 i=0, N2-1
C       xn2(i) = cmplx(5.0-i,0.0)
C 7   continue

C*************************************************************************


      return
      end
```

```
C                      FFT.FOR                   VERSION: 2/03/88
C
C
C  PURPOSE:  THIS PROGRAM COMPUTES THE DISCRETE FOURIER TRANSFORM
C            OR THE INVERSE DISCRETE FOURIER TRANSFORM OF A SET OF
C            COMPLEX INPUT DATA USING A RADIX-2, DECIMATION IN TIME
C            (DIT) FAST FOURIER TRANSFORM (FFT) ALGORITHM.  THE
C            PROGRAM CONSISTS OF A MAIN PROGRAM AND FOUR SUB-
C            ROUTINES. THE SUBROUTINE reversal REARRANGES THE
C            INPUT INTO 'BIT-REVERSED' ORDER; THE SUBROUTINE fft
C            COMPUTES THE FAST FOURIER TRANSFORM; THE SUBROUTINE
C            invfft COMPUTES THE INVERSE FAST FOURIER TRANSFORM; AND
C            THE SUBROUTINE sample ALLOWS THE USER TO GENERATE THE
C            THE INPUT DATA BY WRITING THE APPROPRIATE EQUATIONS.  IF
C            THE USER ELECTS TO GENERATE THE INPUT DATA BY USING THE
C            SUBROUTINE sample, THE EQUATIONS MUST BE WRITTEN INTO
C            THE SUBROUTINE USING STANDARD FORTRAN 77 STATEMENTS AND
C            THE INPUT DATA GENERATED MUST BE STORED IN THE
C            ARRAY xtmp().  THE OUTPUT OF 'FFT.FOR' IS STORED IN
C            THE ARRAY x().  THE USER HAS THE OPTION OF SELECTING
C            ONE OF TWO OPERATING MODES:  BATCH OR TEST.
C            IN BATCH MODE THE AMOUNT OF INTERACTION WITH THE
C            USER IS MINIMIZED AND IT IS ASSUMED THAT THE INPUT
C            PARAMETERS HAVE BEEN STORED IN THE INPUT FILE 'FFT.IN'.
C            IN TEST MODE THE USER IS PROMPTED FOR THE NAME OF
C            THE INPUT FILE OR HAS THE OPTION TO PERFORM A TRIAL
C            RUN USING THE DATA STORED IN THE FILE 'FFT.TST'.
C            IT IS RECOMMENDED THAT FIRST-TIME USERS SELECT TEST
C            MODE AND MAKE A TRIAL RUN WITH THE PRESTORED
C            DATA.  THE TEST MODE ECHOES PORTIONS OF THE INPUT
C            DATA ONTO THE MONITOR TO ALLOW VERIFICATION OF ITS
C            ACCURACY.  THE OUTPUT IS STORED IN TABULAR FORM IN
C            THE FILE 'FFT.OUT' AND IN A FORM SUITABLE FOR PLOTTING
C            IN THE FILE 'FFT.DAT'.
C
C
C******************************   INPUT   ********************************

C
C
C  THIS PROGRAM ASSUMES THAT THERE ARE N = 2**m COMPLEX VALUES IN THE
C  INPUT SEQUENCE.  THE INPUT SEQUENCE IS ASSUMED TO BE DEFINED IN
C  THE INTERVAL: 0 TO N-1. IF THE INPUT SEQUENCE CONSISTS OF 'REAL'
C  NUMBERS THE IMAGINARY PART IS STORED AS 0.0. THE VALUE 'm' AS WELL
C  AS THE OTHER PARAMETERS DESCRIBED BELOW SHOULD BE STORED IN THE
C  INPUT FILE 'FFT.IN'.  ALL OF THE READ STATEMENTS USED BY THIS
C  PROGRAM REQUIRE FORMATTED INPUT.  PARTICULAR ATTENTION SHOULD BE
C  PAID TO THESE FORMATS, ESPECIALLY THE USE OF THE DECIMAL POINT TO
C  DISTINGUISH BETWEEN 'REAL' AND INTEGER DATA.
C
C
```

```
C  NAME           TYPE           RANGE (ARRAYS)         RESTRICTIONS
C  ----           ----           --------------        --------------
C  m             INTEGER                                 0 <= m <= 8
C  dsorce        CHARACTER                                'F' OR 'S'
C  option        CHARACTER                               'FFT' OR 'INV'
C  xtmp()        COMPLEX         0, 1, ..., N-1         1 <= N <= 256
C
C  WHERE:
C
C  m = AN INTEGER THAT SPECIFIES THE NUMBER OF COMPLEX VALUES IN THE
C      INPUT SEQUENCE.  N = 2**m.
C
C  dsorce = A CHARACTER VALUE OF 'F' OR 'S' DENOTING WHETHER THE
C           INPUT DATA IS TO BE READ FROM A FILE (F) OR TO BE
C           GENERATED (S) BY A USER-DEFINED EQUATION LOCATED IN THE
C           SUBROUTINE sample.
C
C  option = A CHARACTER STRING OF THE LETTERS 'FFT' OR 'INV'
C           DENOTING WHETHER THE FFT OR THE INVERSE FFT IS TO BE
C           PERFORMED ON THE INPUT DATA.
C
C  xtmp() = THE ARRAY OF COMPLEX INPUT DATA.  IF dsorce = 'F' IS
C           SELECTED THEN THE USER MUST SUPPLY THE N INPUT VALUES
C           IN THE FILE.  IF dsorce = 'S' THEN THE USER HAS
C           ELECTED TO GENERATE THE INPUT SEQUENCE BY PROVIDING THE
C           APPROPRIATE FORTRAN STATEMENTS IN THE SPACE PROVIDED IN
C           SUBROUTINE sample.  IF THIS METHOD OF DATA GENERATION IS
C           ELECTED THE PROGRAM MUST BE RECOMPILED BEFORE EXECUTION.
C
C  NOTE:  THE INPUT FORMAT STATEMENTS OCCUR IN THE MAIN PROGRAM
C         FOLLOWING THE CAPTION: ********  INPUT FORMAT  ********.
C         THE FORM OF THE INPUT DATA FILE IS:
C
C  LINE#                   ENTRIES                     FORMAT
C  -----                   -------                     ------
C    1                m,dsorce,option             i1,t11,a1,t21,a3
C  2...N+1                 xtmp()                      2f10.0
C
C
C  WHERE:  N = 2**m
C
C  NOTES 1.  LINES 2...N+1 ARE ONLY REQUIRED IF dsorce = 'F'.  IF
C            dsorce = 'S' THEN THE USER HAS ELECTED TO GENERATE THE
C            N = 2**m VALUES FOR xtmp() IN THE SUBROUTINE sample. THE
C            USER MUST PROVIDE THE APPROPRIATE FORTRAN STATEMENTS IN
C            SUBROUTINE sample TO GENERATE THE VALUES FOR xtmp().
C
C        2.  THE FORMAT f10.0 USED FOR INPUT DATA PERMITS THE DECIMAL
C            POINT TO BE PLACED ANYWHERE IN THE FIELD OF TEN COLUMNS
C            AND ALSO ALLOWS THE EXPONENTIAL FORMAT TO BE USED (E.G.,
C            3146.2 = 3.1462E+03).
C
```

```
C
C****************************   OUTPUT   ******************************
C
C
C  THE INPUT DATA AS WELL AS THE OUTPUT DATA ARE STORED IN TABULAR
C  FORM IN THE FILE 'FFT.OUT'. ADDITIONALLY, THE INPUT SEQUENCE (REAL
C  AND IMAGINARY) AND THE OUTPUT SEQUENCE (MAGNITUDE AND PHASE) ARE
C  WRITTEN INTO THE FILE 'FFT.DAT' TO FACILITATE PLOTTING BY A
C  SEPARATE, USER SUPPLIED PROGRAM.  THE FORMAT OF THE DATA IN
C  'FFT.DAT' IS:  e12.6, 2x, e12.6.  THE FIRST ENTRY CORRESPONDS TO
C  THE ORDINATE VALUE AND THE SECOND ENTRY, THE ABSCISSA VALUE.
C  ADDITIONAL HEADER INFORMATION IS WRITTEN INTO 'FFT.DAT' TO ALLOW
C  FOR CONTROL AND LABELING OF EACH PLOT.
C
C
C****************************   EXAMPLE   *****************************
C
C
C  THE INPUT PARAMETERS BELOW ARE STORED IN THE INPUT FILE 'FFT.TST'.
C  THERE ARE EIGHT DATA POINTS IN THE INPUT SEQUENCE AND THE GOAL IS
C  TO COMPUTE THE FAST FOURIER TRANSFORM (FFT) OF THE SEQUENCE.
C  NOTE: N = 2**m = 8  THEREFORE m = 3.
C
C  3         F         FFT
C  0.0       0.0
C  1.0       0.0
C  2.0       0.0
C  3.0       0.0
C  4.0       0.0
C  0.0       0.0
C  0.0       0.0
C  0.0       0.0
C
C
C  THE RESULTING OUTPUT DATA FILE 'FFT.OUT' IS:
C
C  INPUT DATA SOURCEFILE: FFT.TST
C  VALUE OF m = 3     VALUE OF N (2**m) =   8
C  dsorce = F     option = FFT
C
C
C                    INPUT DATA                     INPUT DATA
C                                              (BIT-REVERSED ORDER)
C
C   SAMPLE #   REAL          IMAGINARY      REAL           IMAGINARY
C      0      .000000E+00   .000000E+00    .000000E+00    .000000E+00
C      1      .100000E+01   .000000E+00    .400000E+01    .000000E+00
C      2      .200000E+01   .000000E+00    .200000E+01    .000000E+00
C      3      .300000E+01   .000000E+00    .000000E+00    .000000E+00
C      4      .400000E+01   .000000E+00    .100000E+01    .000000E+00
```

```
C     5      .000000E+00   .000000E+00    .000000E+00    .000000E+00
C     6      .000000E+00   .000000E+00    .300000E+01    .000000E+00
C     7      .000000E+00   .000000E+00    .000000E+00    .000000E+00
C
C
C                                 OUTPUT DATA
C
C   SAMPLE #   REAL           IMAGINARY      MAGNITUDE         PHASE
C                                                            (DEGREES)
C     0       .100000E+02   .000000E+00    .100000E+02   .000000E+00
C     1      -.541421E+01  -.482843E+01    .725448E+01  -.138273E+03
C     2       .200000E+01   .200000E+01    .282843E+01   .450000E+02
C     3      -.258579E+01  -.828427E+00    .271525E+01  -.162236E+03
C     4       .200000E+01   .000000E+00    .200000E+01   .000000E+00
C     5      -.258579E+01   .828427E+00    .271525E+01   .162236E+03
C     6       .200000E+01  -.200000E+01    .282843E+01  -.450000E+02
C     7      -.541421E+01   .482843E+01    .725448E+01   .138273E+03
C
C
C  FOR ILLUSTRATIVE PURPOSES THE INPUT SEQUENCE xtmp() COULD HAVE
C  BEEN GENERATED BY SPECIFYING dsorce = 'S' AND WRITING THE
C  APPROPRIATE FORTRAN STATEMENTS INTO SUBROUTINE sample.  THE
C  STATEMENTS THAT COULD BE USED TO ACCOMPLISH THIS ARE WRITTEN
C  INTO THE SUBROUTINE BUT ARE 'COMMENTED OUT'.
C
C
C***************************  MAIN PROGRAM  ***************************


      character infile*12, option*3, mode*1, dsorce*1
      complex x(0:255), xtmp(0:255)
      real xmag(0:255), xph(0:255), nn

C  PROMPT USER FOR MODE: BATCH OR TEST.

      write(*,1115)
      read(*,1117) mode
      if((mode.eq.'y').or.(mode.eq.'Y')) then
     mode = 'Y'
     write(*,1118)
     read(*,1119) infile
      else
     infile= 'FFT.IN'
      endif

C  UNIT=1 DEFINED AS INPUT FILE.  UNITS=2,3 DEFINED AS OUTPUT FILES.

      open(unit=1,file=infile,status='old',iostat=ierr,err=999)
      open(unit=2,file='FFT.OUT')
      open(unit=3,file='FFT.DAT')
```

```
C  READ INPUT PARAMETERS AND CONDUCT ERROR CHECKS.

      read(1,1000) m, dsorce, option

      if((m.lt.0).or.(m.gt.8)) then
     write(*,1010) m
     stop 'The allowed values for m are:  0 <= m <= 8.'
      endif

      if((option.eq.'FFT').or.(option.eq.'fft')) then
     option = 'FFT'
      elseif((option.eq.'INV').or.(option.eq.'inv')) then
     option = 'INV'
      else
     write(*,1011) option
     stop 'The allowed values for option are: ''FFT'' or ''INV''.'
      endif

      if((dsorce.eq.'F').or.(dsorce.eq.'f')) then
     dsorce = 'F'
      elseif((dsorce.eq.'S').or.(dsorce.eq.'s')) then
     dsorce = 'S'
      else
     write(*,1018) dsorce
     stop 'The allowed values for dsorce are: ''S'' or ''F''.'
      endif

C  DEFINE CONSTANTS.

      N = 2**m
      en = N
      k = 8
      pi = 4.0*atan(1.0)
      numplts = 4

C  FOR dsorce = 'F' READ THE INPUT SEQUENCE FROM THE INPUT FILE.
C  FOR dsorce = 'S' CALL sample TO GENERATE THE INPUT SEQUENCE.
C  THE INPUT SEQUENCE IS STORED IN THE ARRAY xtmp().

      if(dsorce.eq.'F') then
     read(1,1001) (xtmp(i),i=0,N-1)
      else
     call sample(xtmp,N)
      endif

C  FOR TEST MODE ECHO INPUT DATA ONTO THE MONITOR (UNIT = *).

      if(mode.eq.'Y') then
     write(*,1016) infile
     if(N.lt.8) k=N
     write(*,1017) m, N, dsorce, option
     write(*,1012) k
     write(*,1013)
     do 3 i=0, k-1
       write(*,1020) i, xtmp(i)
3        continue
      endif

C  WRITE THE INPUT SEQUENCE INTO FILE: FFT.DAT.

      write(3,2000) numplts
      write(3,2001) N
      write(3,*) 'INPUT SEQUENCE (REAL)'
      write(3,*) 'SAMPLE #'
      write(3,*) 'REAL xtmp() '
      do 55 i=0, N-1
     nn = i
     write(3,2010) nn, real(xtmp(i))
55    continue

      write(3,2001) N
      write(3,*) 'INPUT SEQUENCE (IMAGINARY)'
      write(3,*) 'SAMPLE #'
      write(3,*) 'IMAG xtmp() '
      do 56 i=0, N-1
     nn = i
     write(3,2010) nn, aimag(xtmp(i))
56    continue

C  CALL reversal TO REARRANGE DATA INTO BIT-REVERSED ORDER.

      call reversal(N,m,xtmp,x)

C  WRITE INPUT DATA INTO FILE: FFT.OUT.

      write(2,1016) infile
      write(2,1017) m, N, dsorce, option
      write(2,1014)
      write(2,1015)
      do 8 i=0, N-1
     write(2,1030) i, xtmp(i), x(i)
8     continue

C  CALL fft OR invfft TO PERFORM THE SELECTED COMPUTATION.

      if(option.eq.'INV') then
     call invfft(N,m,x)
      else
     call fft(N,m,x)
      endif
```

```
C  TRANSFORM OUTPUT DATA INTO EXPONENTIAL FORM: xmag*EXP(j*xph).
C  PHASE xph() IS EXPRESSED IN DEGREES.

      do 60 i=0, N-1
     xmag(i) = cabs(x(i))
     if(abs(real(x(i))).lt.1.0e-15) then
       if(abs(aimag(x(i))).le.1.0e-15) xph(i)=0.0
       if(aimag(x(i)).gt.1.0e-15) xph(i)=90.0
       if(aimag(x(i)).lt.-1.0e-15) xph(i)=-90.0
     else
       xph(i)=(180.0/pi)*atan2(aimag(x(i)),real(x(i)))
     endif
60    continue

C  WRITE THE OUTPUT DATA INTO FILE: FFT.DAT.

      write(3,2001) N
      write(3,*) 'OUTPUT MAGNITUDE'
      write(3,*) 'SAMPLE #'
      write(3,*) 'MAGNITUDE'
      do 57 i=0, N-1
     nn = i
     write(3,2010) nn, xmag(i)
57    continue

      write(3,2001) N
      write(3,*) 'OUTPUT PHASE'
      write(3,*) 'SAMPLE #'
      write(3,*) 'PHASE (DEG)'
      do 58 i=0, N-1
     nn = i
     write(3,2010) nn, xph(i)
58    continue

C WRITE THE OUTPUT DATA INTO FILE: FFT.OUT.

      write(2,1025)
      do 5 i=0, N-1
     write(2,1030) i, x(i), xmag(i), xph(i)
5     continue

      write(*,1019)
999   close(unit=1)
      close(unit=2)
      close(unit=3)

      if(ierr.gt.0) then
     write(*,1116) infile, ierr
      endif
```

```
C  ********  INPUT FORMAT  ********

1000  format(i1,t11,a1,t21,a3)
1001  format(2f10.0)

C  ********************************

1010  format(1x,'m = ',i1,2x,'Error, value of m not allowed.')
1011  format(1x,'option = ',a3,2x,'Error, illegal value for option.')
1012 0format(/,' THE FIRST ',i1,' VALUES OF xtmp() ARE LISTED ',
     1'BELOW.',/,' VERIFY THAT THE DATA WAS STORED CORRECTLY.')
1013  format(/,t4,'SAMPLE #',t15,'REAL',t29,'IMAGINARY',/)
1014 0format(///,t25,'INPUT DATA',t57,'INPUT DATA',/,
     1t52,'(BIT-REVERSED ORDER)',/)
1015 0format(t4,'SAMPLE #',t17,'REAL',t33,'IMAGINARY',t49,'REAL',
     1t65,'IMAGINARY')
1016  format(///,' INPUT DATA SOURCEFILE: ',a12)
1017 0format(' VALUE OF m = ',i1,5x,'VALUE OF N (2**m) = ',i3,/,1x,
     1'dsorce = ',a1,5x,'option = ',a3)
1018  format(1x,'dsorce = ',a1,2x,'Error, illegal value for dsorce.')
1019 0format(//,' TABULAR OUTPUT DATA IS STORED IN FILE: FFT.OUT.',
     1/,' PLOTTING DATA IS STORED IN FILE: FFT.DAT.')
1020  format(t7,i1,t13,2(e12.6,2x))
1025 0format(///,t33,'OUTPUT DATA',//,t4,'SAMPLE #',t17,'REAL',
     1t33,'IMAGINARY',t49,'MAGNITUDE',t67,'PHASE',/,t65,'(DEGREES)')
1030  format(t5,i3,t15,4(e12.6,4x))
1115 0format(1x,'DO YOU WISH TO RUN THIS PROGRAM IN TEST',
     1' MODE ?    (Y/N) <CR> : ',\,)
1116 0format(///,1x,'ERROR OPENING INPUT FILE: ',a12,/,1x,'PROGRAM',
     1' TERMINATED.',//,1x,'ERROR CODE:',i4,/////)
1117  format(a1)
1118 0format(/////,1x,'TYPE THE NAME OF YOUR DATA FILE FOLLOWED',
     1' BY <CR>.',/,' IF YOU DESIRE TO MAKE A TEST RUN USING THE',
     2' SAMPLE DATA ALREADY STORED',/,' IN THE FILE: FFT.TST',
     3'         TYPE: FFT.TST <CR>',/,' FILENAME: ',\,)
1119  format(a12)
2000  format(i1)
2001  format(i3)
2010  format(e12.6,2x,e12.6)

      end


C                         SUBROUTINE:  invfft


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY
C            x(), COMPUTES THE INVERSE FAST FOURIER TRANSFORM (IFFT)
C            OF THE ARRAY, AND RETURNS THE RESULTING SEQUENCE IN THE
C            ORIGINAL ARRAY x().
```

```
      subroutine invfft(N,m,x)
      complex x(0:N-1)

      en = N

C  CALCULATE THE COMPLEX CONJUGATE OF THE INPUT SEQUENCE.

      do 70 i=0, N-1
     x(i) = conjg(x(i))
70    continue

C  CALCULATE THE FAST FOURIER TRANSFORM OF THE ARRAY.

      call fft(N,m,x)

C  CALCULATE THE COMPLEX CONJUGATE OF THE RESULTING ARRAY.

      do 80 i=0, N-1
     x(i) = conjg(x(i))/en
80    continue

      return
      end


C                         SUBROUTINE:  reversal


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY
C            CONTAINING THE VALUES xtmp() THAT WERE READ FROM
C            THE INPUT FILE.  THE OUTPUT OF THIS SUBROUTINE IS
C            THE COMPLEX ARRAY x() THAT CONTAINS THE INPUT
C            VALUES IN 'BIT-REVERSED' ORDER.


      subroutine reversal(N,m,xtmp,x)
      complex xtmp(0:N-1), x(0:N-1)

      do 10 k=0, N-1
     newaddr = 0
     maddr = k
     do 20 i=0, m-1
       lrmndr = mod(maddr,2)
       newaddr = newaddr + lrmndr*2**(m-1-i)
       maddr = maddr/2
20      continue
      x(newaddr) = xtmp(k)
10    continue

      return
      end
```

```
C                           SUBROUTINE:  fft


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY x(),
C            COMPUTES THE FAST FOURIER TRANSFORM (FFT) OF THE
C            ARRAY, AND RETURNS THE RESULTING SEQUENCE IN THE
C            ORIGINAL ARRAY x().


      subroutine fft(N,m,x)
      complex x(0:N-1), W, tmp

      pi = 4.0*atan(1.0)
      en = N

      do 50 L=1, m
     ispace = 2**L
     s = N/ispace
     iwidth = ispace/2
     do 40 j=0, iwidth-1
       r = s*j
       alpha = 2.0*pi*r/en
       W = cmplx(cos(alpha),-sin(alpha))
       do 30 itop=j, N-2, ispace
         ibot = itop + iwidth
         tmp = x(ibot)*W
         x(ibot) = x(itop) - tmp
         x(itop) = x(itop) + tmp
30        continue
40      continue
50    continue

      return
      end


C                           SUBROUTINE:  sample


C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE 2**m
C            SAMPLES OF A CONTINUOUS FUNCTION.  THE SAMPLES ARE
C            RETURNED TO THE MAIN PROGRAM IN THE ARRAY xtmp().


      subroutine sample(xtmp,N)
      complex xtmp(0:N-1)

      pi = 4.0*atan(1.0)
      en = N
```

```
C***************************************************************************

C  DEVELOP THE SAMPLING ALGORITHM IN THIS SPACE.  THE STATEMENTS
C  TYPED IN MUST FOLLOW STANDARD FORTRAN 77 RULES AND MAY USE
C  FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS: SIN(), COS(), ABS()...
C  AN EXAMPLE IS SHOWN BELOW.  THE INPUT DATA MUST BE STORED IN
C  THE ARRAY xtmp().  FFT.FOR MUST BE COMPILED AGAIN BEFORE
C  EXECUTION IF THIS SUBROUTINE IS USED.
C
C ***  EXAMPLE  ***
C
C      do 2 i=0, N-1
C        if(i.le.4) then
C          xtmp(i) = cmplx(i,0.0)
C        else
C          xtmp(i) = cmplx(0.0,0.0)
C        endif
C  2   continue

C***************************************************************************


      return
      end
```

```
C                            CONCORFT.FOR         VERSION: 2/03/88
C
C
C  PURPOSE:  THIS PROGRAM PERFORMS ANY ONE OF THE FOLLOWING FOUR
C            COMPUTATIONS GIVEN TWO COMPLEX ARRAYS OF INPUT DATA:
C            LINEAR CONVOLUTION (LCON); LINEAR CORRELATION (LCOR);
C            CIRCULAR CONVOLUTION (CCON); OR CIRCULAR CORRELATION
C            (CCOR) BY USING THE FAST FOURIER TRANSFORM (FFT)
C            ALGORITHM. FOR THE CONVOLUTION OPERATIONS THE PROCEDURE
C            INVOLVES COMPUTING THE FFTs OF THE ARRAYS, MULTIPLYING
C            THE FFTs TOGETHER AND COMPUTING THE INVERSE FFT
C            OF THE RESULT.  FOR THE CORRELATION OPERATIONS THE
C            PROCEDURE IS THE SAME EXCEPT THAT THE CONJUGATE OF
C            THE FFT OF THE FIRST INPUT ARRAY IS MULTIPLIED BY THE
C            FFT OF THE SECOND ARRAY. THE PROGRAM CONSISTS OF A MAIN
C            PROGRAM AND SIX SUBROUTINES. THE SUBROUTINE zeropad
C            EXTENDS THE INPUT ARRAY PASSED TO IT BY ADDING AN
C            APPROPRIATE NUMBER OF ZEROES TO THE ORIGINAL INPUT DATA
C            TO CREATE AN ARRAY OF LENGTH 2**m, m = INTEGER.  THE
C            SUBROUTINE fft COMPUTES THE DISCRETE FOURIER TRANSFORM
C            OF AN ARRAY USING THE RADIX-2 FFT ALGORITHM.  THE
C            SUBROUTINE invfft COMPUTES THE INVERSE DISCRETE
C            FOURIER TRANSFORM OF AN ARRAY USING THE ALTERNATE
C            INVERSION FORMULA.  THE SUBROUTINE reversal REARRANGES
C            THE INPUT DATA INTO BIT-REVERSED ORDER BEFORE fft IS
C            CALLED.  THE SUBROUTINES sampl1 AND sampl2 ALLOW THE
C            USER TO GENERATE EITHER OF THE INPUT ARRAYS BY WRITING
C            THE APPROPRIATE EQUATIONS.  IF THE USER CHOOSES TO
C            GENERATE THE INPUT DATA BY USING EITHER OF THE sampl
C            SUBROUTINE(S), THE EQUATIONS MUST BE WRITTEN INTO
C            THE SUBROUTINE(S) USING STANDARD FORTRAN 77 EXECUTABLE
C            STATEMENTS AND THE VALUES GENERATED MUST BE STORED
C            IN THE ARRAYS xn1() AND xn2().  THE USER HAS THE
C            OPTION OF SELECTING ONE OF TWO OPERATING MODES: BATCH
C            OR TEST.  IN BATCH MODE THE AMOUNT OF INTERACTION
C            WITH THE USER IS MINIMIZED AND IT IS ASSUMED THAT THE
C            INPUT PARAMETERS HAVE BEEN STORED IN THE INPUT FILE
C            'CONCORFT.IN'.  IN TEST MODE THE USER IS PROMPTED
C            FOR THE NAME OF THE INPUT FILE AND HAS THE OPTION
C            TO PERFORM A TRIAL RUN USING THE DATA STORED IN THE
C            FILE 'CONCORFT.TST'.  IT IS RECOMMENDED THAT FIRST-TIME
C            USERS SELECT TEST MODE AND PERFORM A TRIAL RUN
C            WITH THE PRESTORED DATA.  THE TEST MODE ECHOES
C            PORTIONS OF THE INPUT DATA ONTO THE MONITOR TO ALLOW
C            VERIFICATION OF ITS ACCURACY.  THE OUTPUT OF THE
C            PROGRAM 'CONCORFT.FOR' IS STORED IN THE ARRAY xn3()
C            AND IS WRITTEN IN TABULAR FORM INTO THE FILE
C            'CONCORFT.OUT' AND IN A FORM SUITABLE FOR PLOTTING
C            IN THE FILE 'CONCORFT.DAT'.
C
C
```

```
C****************************  INPUT  ********************************
C
C
C  THIS PROGRAM ASSUMES THAT THERE ARE TWO SEQUENCES OF INPUT DATA
C  STORED IN THE ARRAYS xn1() AND xn2() OF LENGTH 'N1' AND 'N2',
C  RESPECTIVELY.  THE SEQUENCES ARE ASSUMED TO BE COMPLEX.  IF THE
C  SEQUENCES CONTAIN REAL VALUES ONLY, THEN THE IMAGINARY PART IS
C  STORED AS 0.0 .  THIS PROGRAM USES A RADIX-2 FFT ALGORITHM.
C  FOR LINEAR CONVOLUTION OR LINEAR CORRELATION (option = LCON,LCOR)
C  THE INPUT ARRAYS DO NOT HAVE TO BE OF LENGTH 2**m, m = INTEGER.
C  THE SUBROUTINE zeropad ADJUSTS THE ARRAY LENGTHS BEFORE
C  THE FFT COMPUTATIONS ARE MADE.  FOR CIRCULAR CONVOLUTION OR
C  CIRCULAR CORRELATION (option = CCON,CCOR) THE ARRAYS MUST BE OF
C  LENGTH 2**m, m = INTEGER BECAUSE EXTENDING THE SEQUENCES BY
C  ZERO PADDING WILL PRODUCE ERRONEOUS RESULTS. THE INPUT SEQUENCES
C  ARE ASSUMED TO BE DEFINED IN THE INTERVALS 0 TO N1-1, AND 0
C  TO N2-1, RESPECTIVELY.  THIS PROGRAM ALLOWS THE USER THE
C  OPTION OF EITHER READING THE INPUT ARRAYS FROM THE DATA
C  FILE OR GENERATING THE INPUT VALUES FROM AN ITERATIVE EQUATION
C  IN THE sampl SUBROUTINE(S).  THE PARAMETERS DESCRIBED
C  BELOW ALLOW THE USER TO SELECT THE DESIRED OPTIONS AND THESE
C  PARAMETERS SHOULD BE STORED IN THE INPUT FILE 'CONCORFT.IN'.
C  ALL OF THE READ STATEMENTS USED BY THIS PROGRAM REQUIRE FORMATTED
C  INPUT.  PARTICULAR ATTENTION SHOULD BE PAID TO THESE FORMATS,
C  ESPECIALLY THE USE OF THE DECIMAL POINT TO DISTINGUISH BETWEEN
C  'REAL' AND INTEGER DATA.
C
C
C  NAME          TYPE       RANGE (ARRAYS)        RESTRICTIONS
C  ----          ----       --------------      ----------------
C  N1           INTEGER                           1 <= N1 <= 128
C  dsrce1       CHARACTER                          'F'  OR  'S'
C  N2           INTEGER                           1 <= N2 <= 128
C  dsrce2       CHARACTER                          'F'  OR  'S'
C  option       CHARACTER                      ONE OF THE FOLLOWING:
C                                            'LCON' 'LCOR' 'CCON' 'CCOR'
C
C  xn1()        COMPLEX      0, 1, ..., N1-1       1 <= N1 <= 128
C  xn2()        COMPLEX      0, 1, ..., N2-1       1 <= N2 <= 128
C
C  WHERE:
C
C  N1 = AN INTEGER THAT SPECIFIES THE NUMBER OF COMPLEX VALUES
C       TO BE STORED IN THE ARRAY xn1().  FOR option = CCON
C       OR CCOR, N1 MUST BE AN INTEGER POWER OF 2, AND N1 AND N2
C       MUST BE EQUAL.
C
C  dsrce1 = A CHARACTER VALUE OF 'F' OR 'S' DENOTING WHETHER THE
C           INPUT ARRAY xn1() IS TO BE READ FROM A FILE (F) OR TO
C           BE GENERATED (S) BY A USER-DEFINED EQUATION LOCATED IN
C           THE SUBROUTINE sampl1.
```

```
C
C  N2 = AN INTEGER THAT SPECIFIES THE NUMBER OF COMPLEX VALUES
C       TO BE STORED IN THE ARRAY xn2().  FOR option = CCON
C       OR CCOR, N2 MUST BE AN INTEGER POWER OF 2, AND N1 AND N2
C       MUST BE EQUAL.
C
C  dsrce2 = A CHARACTER VALUE OF 'F' OR 'S' DENOTING WHETHER THE
C           INPUT ARRAY xn2() IS TO BE READ FROM A FILE (F) OR TO
C           BE GENERATED (S) BY A USER-DEFINED EQUATION LOCATED IN
C           THE SUBROUTINE sampl2.
C
C  option = A CHARACTER STRING OF FOUR LETTERS DENOTING THE
C           COMPUTATION DESIRED:  'LCON' = LINEAR CONVOLUTION
C                                 'LCOR' = LINEAR CORRELATION
C                                 'CCON' = CIRCULAR CONVOLUTION
C                                 'CCOR' = CIRCULAR CORRELATION.
C
C  xn1() = THE FIRST ARRAY OF COMPLEX INPUT DATA. IF dsrce1 = 'F'
C          IS SPECIFIED THE USER MUST SUPPLY THE N1 INPUT VALUES
C          IN THE FILE. IF dsrce1 = 'S' THE USER HAS ELECTED TO
C          GENERATE THE INPUT DATA BY PROVIDING THE APPROPRIATE
C          FORTRAN STATEMENTS IN THE SPACE ALLOCATED IN SUBROUTINE
C          sampl1.  IF THIS METHOD OF DATA GENERATION IS ELECTED
C          THE PROGRAM MUST BE RECOMPILED BEFORE EXECUTION.
C
C  xn2() = THE SECOND ARRAY OF COMPLEX INPUT DATA.  IF dsrce2 =
C          'S' IS SPECIFIED THE USER HAS ELECTED TO PROVIDE THE
C          APPROPRIATE FORTRAN STATEMENTS IN THE SPACE ALLOCATED
C          IN SUBROUTINE sampl2.  IF THIS METHOD OF DATA
C          GENERATION IS ELECTED THE PROGRAM MUST BE RECOMPILED
C          BEFORE EXECUTION.  IF dsrce2 = 'F' THEN THE USER MUST
C          SUPPLY THE N2 INPUT VALUES IN THE FILE.
C
C  NOTE:  THE INPUT FORMAT STATEMENTS OCCUR IN THE MAIN PROGRAM
C         FOLLOWING THE CAPTION: ********  INPUT FORMAT  ********.
C         THE FORM OF THE INPUT DATA FILE IS:
C
C  LINE#              ENTRIES                   FORMAT
C  -----              -------                   ------
C    1               N1,dsrce1                 i3,t11,a1
C    2           N2,dsrce2,option           i3,t11,a1,t21,a4
C  NOTE 1              xn1()                     2f10.0
C  NOTE 2              xn2()                     2f10.0
C
C  NOTES 1.  IF dsrce1 = 'F' THEN THE LINES 3...N1+2 MUST CONTAIN
C            THE VALUES TO BE READ INTO THE ARRAY xn1(). EACH VALUE
C            IS READ AS A COMPLEX NUMBER, I.E.,  REAL   IMAGINARY.
C            IF dsrce1 = 'S' THEN THE USER HAS ELECTED TO GENERATE
C            THE VALUES FOR xn1() IN THE SUBROUTINE sampl1. THE
C            USER MUST PROVIDE THE APPROPRIATE FORTRAN STATEMENTS
C            IN SUBROUTINE sampl1 TO GENERATE xn1().
C
```

```
C        2.  IF dsrce2 = 'F' THEN THE NEXT N2 LINES CONTAIN THE
C            VALUES TO BE READ INTO THE ARRAY xn2().  EACH VALUE
C            IS READ AS A COMPLEX NUMBER, I.E.,  REAL   IMAGINARY.
C            IF dsrce2 = 'S' THEN THE USER HAS ELECTED TO GENERATE
C            THE VALUES FOR xn2() IN THE SUBROUTINE sampl2. THE
C            USER MUST PROVIDE THE APPROPRIATE FORTRAN STATEMENTS
C            IN SUBROUTINE sampl2 TO GENERATE THE ARRAY xn2().
C
C        3.  THE FORMAT 2f10.0 USED FOR INPUT DATA PERMITS THE
C            DECIMAL POINT TO BE PLACED ANYWHERE IN THE FIELD OF TEN
C            COLUMNS AND ALSO ALLOWS THE EXPONENTIAL FORMAT TO BE
C            USED (E.G., 3146.2 = 3.1462E+03).
C
C        4.  IF option = 'CCON' OR 'CCOR' N1 MUST BE EQUAL TO N2.
C
C
C****************************  OUTPUT   *********************************

C
C
C  THE INPUT DATA AS WELL AS THE OUTPUT DATA ARE STORED IN TABULAR
C  FORM IN THE FILE 'CONCORFT.OUT'.  ADDITIONALLY, THE INPUT
C  SEQUENCES AND THE OUTPUT SEQUENCE ARE WRITTEN INTO THE FILE
C  'CONCORFT.DAT' TO FACILITATE PLOTTING BY A SEPARATE, USER
C  SUPPLIED PROGRAM.  THE FORMAT OF THE DATA IN 'CONCORFT.DAT' IS:
C  e12.6, 2x, e12.6.  THE FIRST ENTRY CORRESPONDS TO THE ORDINATE
C  VALUE AND THE SECOND ENTRY, THE ABSCISSA VALUE. ADDITIONAL HEADER
C  INFORMATION IS WRITTEN INTO 'CONCORFT.DAT' TO ALLOW FOR CONTROL
C  AND LABELING OF EACH PLOT.
C
C
C****************************  EXAMPLE  *********************************

C
C
C  THE INPUT PARAMETERS BELOW ARE STORED IN THE INPUT FILE
C  'CONCORFT.TST'.  THE PROGRAM READS THE FIRST 4 VALUES INTO xn1()
C  (dsrce1 = 'F', N1 = 4), AND READS THE NEXT 5 VALUES INTO xn2()
C  (dsrce2 = 'F', N2 = 5).  THE GOAL IS TO CALCULATE THE LINEAR
C  CONVOLUTION OF THE TWO INPUT ARRAYS.
C
C
C  004       F
C  005       F       LCON
C  1.0       0.0
C  1.0       0.0
C  1.0       0.0
C  1.0       0.0
C  2.0       0.0
```

```
C  2.0       0.0
C  2.0       0.0
C  2.0       0.0
C  2.0       0.0
C
C
C  THE RESULTING OUTPUT DATA FILE 'CONCORFT.OUT' IS:
C
C  INPUT DATA SOURCEFILE: CONCORFT.TST
C  N1 =   4     dsrce1 = F          N2 =   5     dsrce2 = F
C  option = LCON
C
C
C                   INPUT DATA
C
C                     xn1()
C     n      REAL              IMAGINARY
C     0     .100000E+01     .000000E+00
C     1     .100000E+01     .000000E+00
C     2     .100000E+01     .000000E+00
C     3     .100000E+01     .000000E+00
C
C                     xn2()
C     n      REAL              IMAGINARY
C     0     .200000E+01     .000000E+00
C     1     .200000E+01     .000000E+00
C     2     .200000E+01     .000000E+00
C     3     .200000E+01     .000000E+00
C     4     .200000E+01     .000000E+00
C
C
C                  OUTPUT DATA
C
C                     xn3()
C     n      REAL              IMAGINARY
C     0     .200000E+01     .894070E-07
C     1     .400000E+01    -.421468E-07
C     2     .600000E+01    -.754979E-07
C     3     .800000E+01    -.168587E-06
C     4     .800000E+01    -.894070E-07
C     5     .600000E+01     .421468E-07
C     6     .400000E+01     .754979E-07
C     7     .200000E+01     .168587E-06
C
C
C  NOTE:   FOR ILLUSTRATIVE PURPOSES THE INPUT SEQUENCES COULD HAVE
C          BEEN GENERATED BY SPECIFYING dsrce# = 'S' AND WRITING THE
C          APPROPRIATE FORTRAN STATEMENTS INTO THE sampl# SUBROUTINES.
C          THE STATEMENTS THAT COULD BE USED TO ACCOMPLISH THIS ARE
C          WRITTEN INTO THE RESPECTIVE SUBROUTINES BUT ARE 'COMMENTED
C          OUT'.
C
```

```
C
C*************************   MAIN PROGRAM   ***************************


      character infile*12, option*4, mode*1, dsrce1*1, dsrce2*1
      character title*20
      complex xn1(0:255), xn2(0:255), xn3(0:255)
      complex xtmp1(0:255), xtmp2(0:255), xtmp3(0:255)
      real nn

C  PROMPT USER FOR MODE: BATCH OR TEST.

      write(*,1115)
      read(*,1117) mode
      if((mode.eq.'y').or.(mode.eq.'Y')) then
     mode = 'Y'
     write(*,1118)
     read(*,1119) infile
      else
     infile = 'CONCORFT.IN'
      endif

C  UNIT=1 DEFINED AS INPUT FILE.  UNITS=2,3 DEFINED AS OUTPUT FILES.

      open(unit=1,file=infile,status='old',iostat=ierr,err=999)
      open(unit=2,file='CONCORFT.OUT')
      open(unit=3,file='CONCORFT.DAT')

C  READ INPUT PARAMETERS AND CONDUCT ERROR CHECKS.

      read(1,1000) N1, dsrce1
      read(1,1001) N2, dsrce2, option

      if((dsrce1.eq.'f').or.(dsrce1.eq.'F')) then
     dsrce1 = 'F'
      elseif((dsrce1.eq.'s').or.(dsrce1.eq.'S')) then
     dsrce1 = 'S'
      else
     write(*,1009) 'dsrce1 = ', dsrce1
     stop 'The allowed values for dsrce1 are: ''F'' or ''S''.'
      endif

      if((dsrce2.eq.'f').or.(dsrce2.eq.'F')) then
     dsrce2 = 'F'
      elseif((dsrce2.eq.'s').or.(dsrce2.eq.'S')) then
     dsrce2 = 'S'
      else
     write(*,1009) 'dsrce2 = ', dsrce2
     stop 'The allowed values for dsrce2 are: ''F'' or ''S''.'
      endif
```

```
       if((option.eq.'ccon').or.(option.eq.'CCON')) then
      option = 'CCON'
      title = 'Circular Convolution'
      N3 = N1
      iend = N3
       elseif((option.eq.'ccor').or.(option.eq.'CCOR')) then
      option = 'CCOR'
      title = 'Circular Correlation'
      N3 = N1
      iend = N3
       elseif((option.eq.'lcon').or.(option.eq.'LCON')) then
      option = 'LCON'
      title = 'Linear Convolution'
       elseif((option.eq.'lcor').or.(option.eq.'LCOR')) then
      option = 'LCOR'
      title = 'Linear Correlation'
       else
      write(*,1011) option
      stop 'The allowed values for option are: CCON,CCOR,LCON,LCOR.'
       endif

       if((N1.lt.1).or.(N1.gt.128)) then
      write(*,1010) 'N1 = ', N1
      stop 'The allowed values for N1 are:  1 <= N1 <= 128.'
       elseif((N2.lt.1).or.(N2.gt.512)) then
      write(*,1010) 'N2 = ', N2
      stop 'The allowed values for N2 are:  1 <= N2 <= 128.'
       elseif((option.eq.'CCON').or.(option.eq.'CCOR')) then
      if(N1.ne.N2) then
        write(*,1008) option, N1, N2
        stop 'For option = ''CCOR'' or ''CCON'' N1 must equal N2.'
      endif
      do 14 m=0, 10
        if(2**m-N3) 14,13,15
15           write(*,1007) option, N1, N2
        stop 'Error, N1 and N2 are not integer powers of 2.'
14       continue
13     endif

C  DEFINE CONSTANTS.

       k = 8
       numplts = 6

C  FOR dsrce# = 'F' READ INPUT SEQUENCE(S) FROM THE DATA FILE.
C  FOR dsrce# = 'S' CALL sample# TO GENERATE THE INPUT SEQUENCE(S).
C  THE INPUT SEQUENCES ARE STORED IN THE ARRAYS xn1(), xn2().
```

```
       if(dsrce1.eq.'F') then
      read(1,1002) (xn1(i),i=0,N1-1)
       else
      call sampl1(xn1,N1)
       endif

       if(dsrce2.eq.'F') then
      read(1,1002) (xn2(i),i=0,N2-1)
       else
      call sampl2(xn2,N2)
       endif


C  FOR TEST MODE ECHO INPUT DATA ONTO MONITOR (UNIT = *).

       if(mode.eq.'Y') then
      write(*,1016) infile
      if((N1.lt.8).or.(N2.lt.8)) k=min(N1,N2)
      write(*,1017) N1, dsrce1, N2, dsrce2
      write(*,1018) option
      write(*,1012) k
      write(*,1013)
      do 3 i=0, k-1
        write(*,1020) i, xn1(i), xn2(i)
3         continue
       endif

C  WRITE THE INPUT SEQUENCES INTO FILE: CONCORFT.DAT.

       write(3,2000) numplts
       write(3,2001) N1
       write(3,*) 'INPUT SEQUENCE xn1 (REAL)'
       write(3,*) 'SAMPLE # (n)'
       write(3,*) 'REAL xn1()'
       do 54 i=0, N1-1
      nn = i
      write(3,2010) nn, real(xn1(i))
54     continue

       write(3,2001) N1
       write(3,*) 'INPUT SEQUENCE xn1 (IMAGINARY)'
       write(3,*) 'SAMPLE # (n)'
       write(3,*) 'IMAG xn1()'
       do 55 i=0, N1-1
      nn = i
      write(3,2010) nn, aimag(xn1(i))
55     continue

       write(3,2001) N2
       write(3,*) 'INPUT SEQUENCE xn2 (REAL)'
       write(3,*) 'SAMPLE # (n)'
       write(3,*) 'REAL xn2()'
      do 56 i=0, N2-1
     nn = i
     write(3,2010) nn, real(xn2(i))
56    continue

      write(3,2001) N2
      write(3,*) 'INPUT SEQUENCE xn2 (IMAGINARY)'
      write(3,*) 'SAMPLE # (n)'
      write(3,*) 'IMAG xn2()'
      do 57 i=0, N2-1
     nn = i
     write(3,2010) nn, aimag(xn2(i))
57    continue

C  WRITE INPUT DATA INTO FILE: CONCORFT.OUT.

      write(2,1016) infile
      write(2,1017) N1, dsrce1, N2, dsrce2
      write(2,1018) option
      write(2,1014)
      write(2,1015) 'xn1()'
      do 65 i=0, N1-1
     write(2,1026) i, xn1(i)
65    continue
      write(2,1015) 'xn2()'
      do 66 i=0, N2-1
     write(2,1026) i, xn2(i)
66    continue

C  FOR LINEAR CONVOLUTION OR LINEAR CORRELATION BOTH INPUT ARRAYS
C  ARE ZERO-PADDED TO LENGTH N3 = 2**m WHERE 2**m IS GREATER THAN
C  OR EQUAL TO N1 + N2 - 1.

      if((option.eq.'LCON').or.(option.eq.'LCOR')) then
     N3 = N1 + N2 - 1
     iend = N3
     do 555 m=0, 10
       if(2**m - N3) 555,556,556
555     continue
556     N3 = 2**m

     call zeropad(xn1,N1,N3)
     call zeropad(xn2,N2,N3)
      endif

C  THE ARRAYS ARE RESEQUENCED IN BIT-REVERSED ORDER BEFORE THE
C  FFT CALCULATION IS PERFORMED.

      call reversal(N3,m,xn1,xtmp1)
      call reversal(N3,m,xn2,xtmp2)
      call fft(N3,m,xtmp1)
      call fft(N3,m,xtmp2)

C  IF option = 'CCON' OR 'LCON' PERFORM CONVOLUTION COMPUTATION.

      if((option.eq.'LCON').or.(option.eq.'CCON')) then
     do 22 i=0, N3-1
       xtmp3(i) = xtmp1(i)*xtmp2(i)
22      continue
      else

C  IF option = 'CCOR' OR 'LCOR' PERFORM CORRELATION COMPUTATION.

     do 75 i=0, N3-1
       xtmp1(i) = conjg(xtmp1(i))
       xtmp3(i) = xtmp1(i)*xtmp2(i)
75      continue

      endif

C  THE RESULTING ARRAYS ARE RESEQUENCED IN BIT-REVERSED ORDER
C  BEFORE THE INVERSE FFT IS CALCULATED.

      call reversal(N3,m,xtmp3,xn3)
      call invfft(N3,m,xn3)

C  WRITE RESULTS INTO FILE: CONCORFT.DAT.

      write(3,2001) iend
      write(3,2003) title
      write(3,*) 'SAMPLE # (n)'
      write(3,*) 'REAL xn3()'
      do 58 i=0, iend-1
     nn = i
     write(3,2010) nn, real(xn3(i))
58    continue

      write(3,2001) iend
      write(3,2003) title
      write(3,*) 'SAMPLE # (n)'
      write(3,*) 'IMAG xn3()'
      do 59 i=0, iend-1
     nn = i
     write(3,2010) nn, aimag(xn3(i))
59    continue
```

```
C  WRITE RESULTS INTO FILE: CONCORFT.OUT.

      write(2,1025)
      write(2,1015) 'xn3()'
      do 67 i=0, iend-1
     write(2,1026) i, xn3(i)
67    continue

      write(*,1019)
999   close(unit=1)
      close(unit=2)
      close(unit=3)
      if(ierr.gt.0) then
     write(*,1116) infile, ierr
      endif

C  ********  INPUT FORMAT  ********

1000  format(i3,t11,a1)
1001  format(i3,t11,a1,t21,a4)
1002  format(2f10.0)

C  ********************************

1007 0format(' option = ',a4,',  N1 = ',i3,',  N2 = ',i3,/,' For ',
     1'option = CCON or CCOR, N1 and N2 must be integer powers of 2.')
1008 0format(' option = ',a4,',  N1 = ',i3,',  N2 = ',i3,',  Error,',
     1' N1 is not equal to N2.')
1009  format(1x,a10,a1,'  Error, value not allowed.')
1010  format(1x,a5,i3,2x,'Error, value not allowed.')
1011  format(1x,'option = ',a4,2x,'Error, illegal value for option.')
1012 0format(/,' THE FIRST ',i1,' VALUES OF INPUT DATA ARE LISTED ',
     1/,' BELOW,  VERIFY THAT THE DATA IS CORRECT.',/)
1013 0format(t21,'xn1()',t53,'xn2()',/,t4,'n',t11,'REAL',t27,
     1'IMAGINARY',t43,'REAL',t59,'IMAGINARY')
1014  format(///,t21,'INPUT DATA',//)
1015  format(/,t21,a7,/,t6,'n',t13,'REAL',t29,'IMAGINARY')
1016  format(//////,' INPUT DATA SOURCEFILE: ',a12)
1017 0format(' N1 = ',i3,5x,'dsrce1 = ',a1,10x,'N2 = ',i3,5x,
     1'dsrce2 = ',a1)
1018  format(1x,'option = ',a4)
1019 0format(/,' TABULAR OUTPUT DATA IS STORED IN FILE: CONCORFT.OUT.'
     1,/,' PLOTTING DATA IS STORED IN FILE: CONCORFT.DAT.')
1020  format(t4,i1,4(4x,e12.6))
1025  format(///,t20,'OUTPUT DATA',//)
1026  format(t4,i3,2(4x,e12.6))
1115 0format(1x,'DO YOU WISH TO RUN THIS PROGRAM IN TEST',
     1' MODE ?    (Y/N) <CR> : ',\,)
1116 0format(///,1x,'ERROR OPENING INPUT FILE: ',a12,/,1x,'PROGRAM',
     1' TERMINATED.',//,1x,'ERROR CODE:',i4,/////)
1117  format(a1)
```

```
1118 0format(/////,1x,'TYPE THE NAME OF YOUR DATA FILE FOLLOWED',
    1' BY <CR>.',/,' IF YOU DESIRE TO MAKE A TEST RUN USING THE',
    2' SAMPLE DATA ALREADY STORED',/,' IN THE FILE: CONCORFT.TST',
    3'        TYPE: CONCORFT.TST <CR>',/,' FILENAME: ',\,)
1119  format(a12)
2000  format(i1)
2001  format(i3)
2003  format(a20)
2010  format(e12.6,2x,e12.6)

      end


C                          SUBROUTINE: zeropad


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY xn()
C            OF LENGTH N, AND ZERO PADS THE ARRAY TO LENGTH N3 WHERE
C            N3 = N1 + N2 - 1.


      subroutine zeropad(xn,N,N3)
      complex xn(0:N3-1)

      do 33 i=N, N3-1
     xn(i) = cmplx(0.0,0.0)
33    continue

      return
      end


C                          SUBROUTINE: invfft


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY
C            x(), COMPUTES THE INVERSE FFT OF THE ARRAY, AND
C            RETURNS THE RESULTING SEQUENCE IN THE ARRAY x().


      subroutine invfft(N,m,x)
      complex x(0:N-1)

      en = N

C  COMPUTE THE COMPLEX CONJUGATE OF THE INPUT DATA.

      do 70 i=0, N-1
     x(i) = conjg(x(i))
70    continue
```

```
C  COMPUTE THE FAST FOURIER TRANSFORM OF THE ARRAY.

      call fft(N,m,x)

C  COMPUTE THE COMPLEX CONJUGATE OF THE RESULTING ARRAY.

      do 80 i=0, N-1
     x(i) = conjg(x(i))/en
80    continue

      return
      end


C                     SUBROUTINE: reversal


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY
C            xtmp(). THE OUTPUT OF THIS SUBROUTINE IS THE COMPLEX
C            ARRAY x() THAT CONTAINS THE INPUT VALUES IN BIT-
C            REVERSED ORDER.


      subroutine reversal(N,m,xtmp,x)
      complex xtmp(0:N-1), x(0:N-1)

      do 10 k=0, N-1
     newaddr = 0
     maddr = k
     do 20 i=0, m-1
       lrmndr = mod(maddr,2)
       newaddr = newaddr + lrmndr*2**(m-1-i)
       maddr = maddr/2
20      continue
      x(newaddr) = xtmp(k)
10    continue

      return
      end


C                     SUBROUTINE: fft


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE COMPLEX ARRAY x(),
C            COMPUTES THE FAST FOURIER TRANSFORM (FFT) OF THE
C            ARRAY, AND RETURNS THE RESULTING SEQUENCE IN THE
C            ORIGINAL ARRAY x().


      subroutine fft(N,m,x)
      complex x(0:N-1), W, tmp
      pi = 4.0*atan(1.0)
      en = N

      do 50 L=1, m
     ispace = 2**L
     s = N/ispace
     iwidth = ispace/2
     do 40 j=0, iwidth-1
       r = s*j
       alpha = 2.0*pi*r/en
       W = cmplx(cos(alpha),-sin(alpha))
       do 30 itop=j, N-2, ispace
         ibot = itop + iwidth
         tmp = x(ibot)*W
         x(ibot) = x(itop) - tmp
         x(itop) = x(itop) + tmp
30        continue
40      continue
50    continue

      return
      end


C                       SUBROUTINE: sampl1


C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE SAMPLES
C            OF A CONTINUOUS FUNCTION AND STORE THEM IN THE ARRAY
C            xn1(). IF dsrce1 = 'S' THEN THE MAIN PROGRAM WILL
C            CALL THIS SUBROUTINE TO GENERATE THE VALUES FOR
C            xn1(). IF dsrce1 DOES NOT EQUAL 'S' THEN THIS
C            SUBROUTINE WILL NOT BE CALLED BY THE MAIN PROGRAM.


      subroutine sampl1(xn1,N1)
      complex xn1(0:N1-1)

      pi = 4.0*atan(1.0)
      en1 = N1

C**********************************************************************

C  DEVELOP THE SAMPLING ALGORITHM FOR xn1() IN THIS SPACE. THE
C  STATEMENTS TYPED IN MUST FOLLOW STANDARD FORTRAN 77 RULES AND
C  MAY USE FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS: SIN(),COS()...
C  AN EXAMPLE OF AN ALGORITHM GENERATING VALUES FOR xn1() IS:
C
C ***  EXAMPLE  ***
C
```

```
C     do 6 i=0, N1-1
C       xn1(i) = cmplx(1.0,0.0)
C 6   continue



C************************************************************************


      return
      end


C                              SUBROUTINE: sampl2


C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE SAMPLES
C            OF A CONTINUOUS FUNCTION AND STORE THEM IN THE ARRAY
C            xn2(). IF dsrce2 = 'S' THEN THE MAIN PROGRAM WILL
C            CALL THIS SUBROUTINE TO GENERATE THE VALUES FOR
C            xn2().  IF dsrce2 DOES NOT EQUAL 'S' THEN THIS
C            SUBROUTINE WILL NOT BE CALLED BY THE MAIN PROGRAM.


      subroutine sampl2(xn2,N2)
      complex xn2(0:N2-1)

      pi = 4.0*atan(1.0)
      en2 = N2

C************************************************************************

C  DEVELOP THE SAMPLING ALGORITHM FOR xn2() IN THIS SPACE. THE
C  STATEMENTS TYPED IN MUST FOLLOW STANDARD FORTRAN 77 RULES AND
C  MAY USE FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS: SIN(),COS()...
C  AN EXAMPLE OF AN ALGORITHM GENERATING VALUES FOR xn2() IS:
C
C ***  EXAMPLE  ***
C
C     do 7 i=0, N2-1
C       xn2(i) = cmplx(2.0,0.0)
C 7   continue

C************************************************************************


      return
      end
```

```
C                          CONCOR.FOR           VERSION:  2/03/88
C
C
C  PURPOSE:  THIS PROGRAM PERFORMS EITHER THE LINEAR CONVOLUTION
C            (LCON) OR THE LINEAR CORRELATION (LCOR) OF TWO ARRAYS
C            OF INPUT DATA.  THE PROGRAM CONSISTS OF A MAIN PROGRAM
C            AND FOUR SUBROUTINES.  THE SUBROUTINE convol PERFORMS
C            THE CONVOLUTION OF THE TWO INPUT ARRAYS xn1() and xn2()
C            AND STORES THE RESULTS IN THE OUTPUT ARRAY yn().  THE
C            SUBROUTINE correl PERFORMS THE CORRELATION OF THE TWO
C            ARRAYS xn1() AND xn2() ACCORDING TO THE EQUATION:
C            Rxn1xn2(p) = SUM[xn1(m)*xn2(m+p)].  THE TWO SUBROUTINES
C            sampl1 AND SAMPL2 ALLOW THE USER THE OPTION OF
C            GENERATING EITHER OF THE TWO INPUT ARRAYS BY WRITING
C            THE APPROPRIATE EQUATIONS.  IF THE USER CHOOSES TO
C            GENERATE THE INPUT DATA BY USING EITHER OF THE sampl
C            SUBROUTINES, THE EQUATIONS MUST BE WRITTEN INTO THE
C            SUBROUTINES USING STANDARD FORTRAN 77 EXECUTABLE STATE-
C            MENTS AND THE VALUES GENERATED MUST BE STORED IN THE
C            ARRAYS xn1() AND xn2().  THE USER HAS THE OPTION OF
C            SELECTING ONE OF TWO OPERATING MODES: BATCH OR TEST. IN
C            BATCH MODE THE AMOUNT OF INTERACTION WITH THE USER IS
C            MINIMIZED AND IT IS ASSUMED THAT THE INPUT PARAMETERS
C            HAVE BEEN STORED IN THE INPUT FILE 'CONCOR.IN'.  IN
C            TEST MODE THE USER IS PROMPTED FOR THE NAME OF THE
C            INPUT FILE AND HAS THE OPTION TO PERFORM A TRIAL RUN
C            USING THE DATA STORED IN THE FILE 'CONCOR.TST'.
C            IT IS RECOMMENDED THAT FIRST-TIME USERS SELECT THE
C            TEST MODE AND PERFORM A TRIAL RUN WITH THE PRESTORED
C            DATA.  THE TEST MODE ECHOES PORTIONS OF THE INPUT
C            DATA ONTO THE MONITOR TO ALLOW VERIFICATION OF ITS
C            ACCURACY.  THE OUTPUT OF THE PROGRAM 'CONCOR.FOR' IS
C            STORED IN THE ARRAY yn() IF LINEAR CONVOLUTION (LCON)
C            IS SELECTED OR IN THE ARRAY R() IF LINEAR CORRELATION
C            (LCOR) IS SELECTED.  THE OUTPUT IS STORED IN TABULAR
C            FORM IN THE FILE 'CONCOR.OUT' AND IN A FORM SUITABLE
C            FOR PLOTTING IN THE FILE 'CONCOR.DAT'.
C
C
C********************************   INPUT   ********************************

C
C
C  THIS PROGRAM ASSUMES THAT THERE ARE TWO SEQUENCES OF INPUT DATA
C  STORED IN THE ARRAYS xn1() AND xn2().  THE SEQUENCE xn1() EXISTS
C  IN THE RANGE: ns1 <= n <= ne1.  THE SEQUENCE xn2() EXISTS IN THE
C  RANGE: ns2 <= n <= ne2.  THE CONSTRAINTS ON THESE VALUES ARE:
C  -128 <= ns1 <= ne1 <= 128  AND  -128 <= ns2 <= ne2 <= 128.
C  THIS PROGRAM ALLOWS THE USER THE OPTION OF EITHER READING
C  THE INPUT ARRAYS FROM A DATA FILE OR OF GENERATING THE INPUT
C  VALUES FROM AN ITERATIVE EQUATION IN THE sampl SUBROUTINE(S).
C  THE PARAMETERS DESCRIBED BELOW ALLOW THE USER TO SELECT THE
```

```
C  DESIRED OPTIONS AND THESE PARAMETERS MUST BE STORED IN THE INPUT
C  FILE 'CONCOR.IN'.  ALL OF THE READ STATEMENTS USED BY THIS
C  PROGRAM REQUIRE FORMATTED INPUT.  PARTICULAR ATTENTION SHOULD BE
C  PAID TO THESE FORMATS, ESPECIALLY THE USE OF THE DECIMAL POINT TO
C  DISTINGUISH BETWEEN 'REAL' AND INTEGER DATA.
C
C
C  NAME          TYPE        RANGE (ARRAYS)          RESTRICTIONS
C  ----          ----        --------------        -----------------
C  option       CHARACTER                          'LCON'  OR  'LCOR'
C  ns1          INTEGER                            -128 <= ns1 <= 128
C  ne1          INTEGER                            -128 <= ne1 <= 128
C  dsrce1       CHARACTER                             'F'  OR  'S'
C  ns2          INTEGER                            -128 <= ns2 <= 128
C  ne2          INTEGER                            -128 <= ne2 <= 128
C  dsrce2       CHARACTER                             'F'  OR  'S'
C  xn1(n)       REAL        ns1 <= n <= ne1          ns1 <= ne1
C  xn2(n)       REAL        ns2 <= n <= ne2          ns2 <= ne2
C
C  WHERE:
C
C  option =  A CHARACTER STRING OF FOUR LETTERS DENOTING THE
C            COMPUTATION DESIRED:  'LCON' = LINEAR CONVOLUTION
C                                  'LCOR' = LINEAR CORRELATION.
C
C  ns1 = AN INTEGER VALUE THAT SPECIFIES THE STARTING SAMPLE POINT OF
C        THE SEQUENCE xn1().
C
C  ne1 = AN INTEGER VALUE THAT SPECIFIES THE ENDING SAMPLE POINT OF
C        THE SEQUENCE xn1().
C
C  dsrce1 =  A CHARACTER VALUE OF 'F' OR 'S' DENOTING WHETHER THE
C            INPUT ARRAY xn1() IS TO BE READ FROM A FILE (F) OR TO
C            BE GENERATED (S) BY A USER-DEFINED EQUATION LOCATED IN
C            THE SUBROUTINE sampl1.
C
C  ns2 = AN INTEGER VALUE THAT SPECIFIES THE STARTING SAMPLE POINT OF
C        THE SEQUENCE xn2().
C
C  ne2 = AN INTEGER VALUE THAT SPECIFIES THE ENDING SAMPLE POINT OF
C        THE SEQUENCE xn2().
C
C  dsrce2 =  A CHARACTER VALUE OF 'F' OR 'S' DENOTING WHETHER THE
C            INPUT ARRAY xn2() IS TO BE READ FROM A FILE (F) OR TO
C            BE GENERATED (S) BY A USER-DEFINED EQUATION LOCATED IN
C            THE SUBROUTINE sampl2.
C
C  xn1() =  THE FIRST ARRAY OF INPUT DATA. IF dsrce1 = 'F' IS
C           SPECIFIED, THE USER MUST SUPPLY THE N1 INPUT VALUES IN
C           THE FILE (WHERE N1 = ne1 - ns1 + 1).  IF dsrce1 = 'S'
C           THEN THE USER HAS ELECTED TO GENERATE THE INPUT
C           SEQUENCE xn1() BY WRITING THE APPROPRIATE FORTRAN
```

```
C            STATEMENTS IN THE SPACE ALLOCATED IN SUBROUTINE sampl1.
C            IF THIS METHOD OF DATA GENERATION IS ELECTED THE PROGRAM
C            MUST BE RECOMPILED BEFORE EXECUTION.
C
C  xn2() =   THE SECOND ARRAY OF INPUT DATA. IF dsrce2 = 'F' IS
C            SPECIFIED, THE USER MUST SUPPLY THE N2 INPUT VALUES IN
C            THE FILE (WHERE N2 = ne2 - ns2 + 1).  IF dsrce2 = 'S'
C            THEN THE USER HAS ELECTED TO GENERATE THE INPUT
C            SEQUENCE xn2() BY WRITING THE APPROPRIATE FORTRAN
C            STATEMENTS IN THE SPACE ALLOCATED IN SUBROUTINE sampl2.
C            IF THIS METHOD OF DATA GENERATION IS ELECTED THE PROGRAM
C            MUST BE RECOMPILED BEFORE EXECUTION.
C
C  NOTE:   THE INPUT FORMAT STATEMENTS OCCUR IN THE MAIN PROGRAM
C          FOLLOWING THE CAPTION: ********  INPUT FORMAT  ********.
C          THE FORM OF THE INPUT DATA FILE IS:
C
C  LINE#             ENTRIES                      FORMAT
C  -----             -------                      ------
C   1                option                         a4
C   2            ns1,ne1,dsrce1              i4,t11,i4,t21,a1
C   3            ns2,ne2,dsrce2              i4,t11,i4,t21,a1
C  NOTE1             xn1()                         f10.0
C  NOTE2             xn2()                         f10.0
C
C  NOTES 1.   IF dsrce1 = 'F' THEN THE NEXT N1 LINES MUST CONTAIN
C             THE VALUES TO BE READ INTO THE ARRAY xn1().
C             IF dsrce1 = 'S' THEN THE USER HAS ELECTED TO GENERATE
C             THE VALUES FOR xn1() IN THE SUBROUTINE sampl1.  THE
C             USER MUST PROVIDE THE APPROPRIATE FORTRAN STATEMENTS
C             IN SUBROUTINE sampl1 TO GENERATE xn1().
C
C        2.   IF dsrce2 = 'F' THEN THE NEXT N2 LINES CONTAIN THE
C             VALUES TO BE READ INTO THE ARRAY xn2().
C             IF dsrce2 = 'S' THEN THE USER HAS ELECTED TO GENERATE
C             THE VALUES FOR xn2() IN THE SUBROUTINE sampl2.  THE
C             USER MUST PROVIDE THE APPROPRIATE FORTRAN STATEMENTS
C             IN SUBROUTINE sampl2 TO GENERATE THE ARRAY xn2().
C
C        3.   THE FORMAT f10.0 USED FOR INPUT DATA PERMITS THE
C             DECIMAL POINT TO BE PLACED ANYWHERE IN THE FIELD OF TEN
C             COLUMNS AND ALSO ALLOWS THE EXPONENTIAL FORMAT TO BE
C             USED (E.G., 3146.2 = 3.1462E+03).
C
C
C******************************   OUTPUT   ******************************

C
C
C  THE OUTPUT SEQUENCE GENERATED BY THE PROGRAM WILL EXIST ONLY OVER
C  THE NON-ZERO RANGE DETERMINED AS FOLLOWS:  FOR option = 'LCON'
```

```
C  yn(n) EXISTS IN THE RANGE ns1 + ns2 <= n <= ne1 + ne2; FOR option =
C  'LCOR' R(p) EXISTS IN THE RANGE  ns2 - ne1 <= p <= ne2 - ns1.  THE
C  INPUT DATA AS WELL AS THE OUTPUT DATA ARE STORED IN TABULAR FORM
C  IN THE FILE 'CONCOR.OUT'.  ADDITIONALLY, THE INPUT SEQUENCES AND
C  THE OUTPUT SEQUENCE ARE WRITTEN INTO THE FILE 'CONCOR.DAT' TO
C  FACILITATE PLOTTING BY A SEPARATE, USER SUPPLIED PROGRAM.  THE
C  FORMAT OF THE DATA IN 'CONCOR.DAT' IS:  e12.6, 2x, e12.6.  THE
C  FIRST ENTRY CORRESPONDS TO THE ORDINATE VALUE AND THE SECOND ENTRY,
C  THE ABSCISSA VALUE.  ADDITIONAL HEADER INFORMATION IS WRITTEN INTO
C  'CONCOR.DAT' TO ALLOW FOR CONTROL AND LABELING OF EACH PLOT.
C
C
C*******************************   EXAMPLE   ******************************

C
C
C  THE INPUT PARAMETERS BELOW ARE STORED IN THE INPUT FILE
C  'CONCOR.TST' .  THE PROGRAM READS THE FIRST 4 VALUES INTO xn1()
C  (dsrce1 = 'F'), AND READS THE NEXT 5 VALUES INTO xn2() (dsrce2 =
C  'F').  THE GOAL IS TO COMPUTE THE LINEAR CONVOLUTION OF THE
C  TWO INPUT ARRAYS.
C
C  THE SEQUENCE xn1() EXTENDS FROM -3 TO 0 (ns1 = -3, ne1 = 0).
C  xn1(n) = 1.0  FOR  ns1 <= n <= ne1
C         = 0.0  OTHERWISE
C
C  THE SEQUENCE xn2() EXTENDS FROM 0 TO 4 (ns2 = 0, ne2 = 4).
C  xn2(n) = n+1  FOR ns2 <= n <= ne2
C         = 0.0  OTHERWISE
C
C  THE APPROPRIATE INPUT FILE ENTRIES ARE:
C
C  LCON
C    -3      0000      F
C  0000      0004      F
C  1.0
C  1.0
C  1.0
C  1.0
C  1.0
C  2.0
C  3.0
C  4.0
C  5.0
C
C
C  THE RESULTING OUTPUT DATA FILE 'CONCOR.OUT' IS:
C
C  INPUT DATA SOURCEFILE: CONCOR.TST
C  ns1 =   -3   ne1 =    0   dsrce1 = F
C  ns2 =    0   ne2 =    4   dsrce2 = F
C  option = LCON
```

```
C
C     INPUT DATA
C
C      n       xn1(n)
C
C     -3     .100000E+01
C     -2     .100000E+01
C     -1     .100000E+01
C      0     .100000E+01
C
C      n       xn2(n)
C
C      0     .100000E+01
C      1     .200000E+01
C      2     .300000E+01
C      3     .400000E+01
C      4     .500000E+01
C
C
C     OUTPUT DATA
C
C      n        yn(n)
C
C     -3     .100000E+01
C     -2     .300000E+01
C     -1     .600000E+01
C      0     .100000E+02
C      1     .140000E+02
C      2     .120000E+02
C      3     .900000E+01
C      4     .500000E+01
C
C
C NOTE:  FOR ILLUSTRATIVE PURPOSES THE INPUT SEQUENCES xn1()
C        AND xn2() COULD HAVE BEEN GENERATED BY WRITING THE
C        APPROPRIATE STATEMENTS IN SUBROUTINES sampl1 AND
C        AND sampl2.  THE STATEMENTS THAT COULD BE USED TO
C        ACCOMPLISH THIS ARE WRITTEN INTO THE SUBROUTINES BUT
C        ARE 'COMMENTED OUT'.
C
C
C***************************  MAIN PROGRAM  ***************************


      character infile*12, option*4, mode*1, dsrce1*1, dsrce2*1
      character ylabl*5, title*18, xlabl*12
      real xn1(-128:128) ,xn2(-128:128), yn(-256:256), R(-256:256)
      real nn
      integer p
```

```
C  PROMPT USER FOR MODE: BATCH OR TEST.

      write(*,1115)
      read(*,1117) mode
      if((mode.eq.'y').or.(mode.eq.'Y')) then
     mode = 'Y'
     write(*,1118)
     read(*,1119) infile
      else
     infile = 'CONCOR.IN'
      endif

C  UNIT=1 DEFINED AS INPUT FILE.  UNITS=2,3 DEFINED AS OUTPUT FILES.

      open(unit=1,file=infile,status='old',iostat=ierr,err=999)
      open(unit=2,file='CONCOR.OUT')
      open(unit=3,file='CONCOR.DAT')

C  READ INPUT PARAMETERS AND CONDUCT ERROR CHECKS.

      read(1,1000) option
      read(1,1001) ns1, ne1, dsrce1
      read(1,1001) ns2, ne2, dsrce2

      if((option.eq.'lcon').or.(option.eq.'LCON')) then
     option = 'LCON'
     ylabl = 'yn(n)'
     xlabl = 'SAMPLE # (n)'
     title = 'Linear Convolution'
     ns3 = ns1 + ns2
     ne3 = ne1 + ne2
      elseif((option.eq.'lcor').or.(option.eq.'LCOR')) then
     option = 'LCOR'
     ylabl = 'R(p)'
     xlabl = 'SAMPLE # (p)'
     title = 'Linear Correlation'
     ns3 = ns2 - ne1
     ne3 = ne2 - ns1
      else
     write(*,1011) option
     stop 'The allowed values for option are: ''LCON'' or ''LCOR''.'
      endif

      if((ns1.lt.-128).or.(ns1.gt.128)) then
     write(*,1010) 'ns1 = ', ns1
     stop 'The allowed values for ns1 are: -128 <= ns1 <= 128.'
      elseif((ns2.lt.-128).or.(ns2.gt.128)) then
     write(*,1010) 'ns2 = ', ns2
     stop 'The allowed values for ns2 are: -128 <= ns2 <= 128.'
      endif
      if((ne1.lt.-128).or.(ne1.gt.128)) then
     write(*,1010) 'ne1 = ', ne1
     stop 'The allowed values for ne1 are: -128 <= ne1 <= 128.'
      elseif((ne2.lt.-128).or.(ne2.gt.128)) then
     write(*,1010) 'ne2 = ', ne2
     stop 'The allowed values for ne2 are: -128 <= ne2 <= 128.'
      endif

      if(ne1.lt.ns1) then
     write(*,1120) 'ns1 = ', ns1, 'ne1 = ', ne1
     stop 'The value ne1 must be greater than or equal to ns1.'
      endif

      if(ne2.lt.ns2) then
     write(*,1120) 'ns2 = ', ns2, 'ne2 = ', ne2
     stop 'The value ne2 must be greater than or equal to ns2.'
      endif


      if((dsrce1.eq.'f').or.(dsrce1.eq.'F')) then
     dsrce1 = 'F'
      elseif((dsrce1.eq.'s').or.(dsrce1.eq.'S')) then
     dsrce1 = 'S'
      else
     write(*,1009) 'dsrce1 = ', dsrce1
     stop 'The allowed values for dsrce1 are: ''F'' or ''S''.'
      endif

      if((dsrce2.eq.'f').or.(dsrce2.eq.'F')) then
     dsrce2 = 'F'
      elseif((dsrce2.eq.'s').or.(dsrce2.eq.'S')) then
     dsrce2 = 'S'
      else
     write(*,1009) 'dsrce2 = ', dsrce2
     stop 'The allowed values for dsrce2 are: ''F'' or ''S''.'
      endif


C  DEFINE CONSTANTS ACCORDING TO THE FOLLOWING SCHEME:

C  N1 = THE NUMBER OF SAMPLES IN THE SEQUENCE xn1().
C  N2 = THE NUMBER OF SAMPLES IN THE SEQUENCE xn2().
C  N3 = THE NUMBER OF SAMPLES IN THE OUTPUT SEQUENCE.
C  k = A DUMMY VARIABLE USED FOR WRITING THE OUTPUT TO THE MONITOR.
C  numplts = A CONTROL PARAMETER FOR THE DATA STORED IN 'CONCOR.DAT'.

      N1 = ne1 - ns1 + 1
      N2 = ne2 - ns2 + 1
      N3 = ne3 - ns3 + 1
      k = 8
      numplts = 3
```

```
C  FOR dsrce# = 'F' READ INPUT DATA FROM THE DATA FILE.
C  FOR dsrce# = 'S' CALL sampl# TO GENERATE THE INPUT DATA.
C  THE INPUT DATA IS STORED IN THE ARRAYS xn1(), xn2().

       if(dsrce1.eq.'F') then
      read(1,1002) (xn1(i),i=ns1,ne1)
       else
      call sampl1(ns1,ne1,xn1)
       endif

       if(dsrce2.eq.'F') then
      read(1,1002) (xn2(i),i=ns2,ne2)
       else
      call sampl2(ns2,ne2,xn2)
       endif


C  FOR TEST MODE ECHO INPUT DATA ONTO MONITOR (UNIT = *).

       if(mode.eq.'Y') then
      write(*,1016) infile
      if((N1.lt.8).or.(N2.lt.8)) k=min(N1,N2)
      write(*,1017) 'ns1 = ',ns1, 'ne1 = ',ne1, 'dsrce1 = ',dsrce1
      write(*,1017) 'ns2 = ',ns2, 'ne2 = ',ne2, 'dsrce2 = ',dsrce2
      write(*,1018) option
      write(*,1012) k
      write(*,1013)
      indx1 = ns1
      indx2 = ns2
      do 3 i=0, k-1
        write(*,1020) indx1, xn1(indx1), indx2, xn2(indx2)
        indx1 = indx1+1
        indx2 = indx2+1
3         continue
       endif

C  WRITE THE INPUT SEQUENCES INTO FILE: CONCOR.DAT.

       write(3,2000) numplts
       write(3,2001) N1
       write(3,*) 'INPUT SEQUENCE xn1(n)'
       write(3,*) 'SAMPLE # (n)'
       write(3,*) 'xn1(n)'
       do 55 n=ns1, ne1
      nn = n
      write(3,2010) nn, xn1(n)
55     continue

       write(3,2001) N2
       write(3,*) 'INPUT SEQUENCE xn2(n)'
       write(3,*) 'SAMPLE # (n)'
       write(3,*) 'xn2(n)'
      do 56 n=ns2, ne2
     nn = n
     write(3,2010) nn, xn2(n)
56    continue

C  WRITE INPUT DATA INTO FILE: CONCOR.OUT.

      write(2,1016) infile
      write(2,1017) 'ns1 = ',ns1, 'ne1 = ',ne1, 'dsrce1 = ',dsrce1
      write(2,1017) 'ns2 = ',ns2, 'ne2 = ',ne2, 'dsrce2 = ',dsrce2
      write(2,1018) option
      write(2,1025) 'INPUT'
      write(2,1015) 'n', 'xn1(n)'
      do 4 n=ns1, ne1
     write(2,1026) n, xn1(n)
4     continue

      write(2,1015) 'n', 'xn2(n)'
      do 5 n=ns2, ne2
     write(2,1026) n, xn2(n)
5     continue

C  IF option = 'LCON' CALL convol TO PERFORM CONVOLUTION COMPUTATION.

      if(option.eq.'LCON') then
     call convol(ns1,N1,ns2,N2,ns3,ne3,xn1,xn2,yn)

C  IF option = 'LCOR' CALL correl TO PERFORM CORRELATION COMPUTATION.

      else
     call correl(ns1,ne1,ns2,ne2,ns3,ne3,xn1,xn2,R)
      endif

C  WRITE RESULTS INTO FILE: CONCOR.DAT.

      write(3,2001) N3
      write(3,2003) title
      write(3,2004) xlabl
      write(3,2005) ylabl
     do 57 n=ns3, ne3
       nn = n
       if(option.eq.'LCON') then
         write(3,2010) nn, yn(n)
       else
         write(3,2010) nn, R(n)
       endif
57     continue
```

```
C  WRITE RESULTS INTO FILE: CONCOR.OUT.

      if(option.eq.'LCON') then
     write(2,1025) 'OUTPUT'
     write(2,1015) 'n', ylabl
     do 9 n=ns3, ne3
       write(2,1026) n, yn(n)
9        continue

      else
     write(2,1025) 'OUTPUT'
     write(2,1015) 'p', ylabl
     do 11 p=ns3, ne3
       write(2,1026) p, R(p)
11       continue
      endif

      write(*,1019)
999   close(unit=1)
      close(unit=2)
      close(unit=3)

      if(ierr.gt.0) then
     write(*,1116) infile, ierr
      endif

C  ********  INPUT FORMAT  ********

1000  format(a4)
1001  format(i4,t11,i4,t21,a1)
1002  format(f10.0)

C  ********************************

1009  format(1x,a10,a1,'  Error, value not allowed.')
1010  format(1x,a6,i4,2x,'Error, value not allowed.')
1011  format(1x,'option = ',a4,2x,'Error, illegal value for option.')
1012 0format(/,' THE FIRST ',i1,' VALUES OF INPUT DATA ARE LISTED ',
     1/,' BELOW,  VERIFY THAT THE DATA IS CORRECT.',/)
1013  format(t7,'n',t12,'xn1()',t28,'n',t33,'xn2()')
1015  format(/,t7,a1,t14,a6,/)
1016  format(///,' INPUT DATA SOURCEFILE: ',a12)
1017  format(1x,a6,i4,3x,a6,i4,3x,a9,a1)
1018  format(1x,'option = ',a4)
1019 0format(//,' TABULAR OUTPUT DATA IS STORED IN FILE: CONCOR.OUT.',
     1/,' PLOTTING DATA IS STORED IN FILE: CONCOR.DAT.')
1020  format(t4,i4,t10,e10.4,t25,i4,t31,e10.4)
1025  format(///,t7,a6,' DATA',/)
1026  format(t4,i4,t12,e12.6)
1115 0format(1x,'DO YOU WISH TO RUN THIS PROGRAM IN TEST',
     1' MODE ?    (Y/N) <CR> : ',\,)
```

```
1116 0format(///,1x,'ERROR OPENING INPUT FILE: ',a12,/,1x,'PROGRAM',
     1' TERMINATED.',//,1x,'ERROR CODE:',i4,/////)
1117  format(a1)
1118 0format(/////,1x,'TYPE THE NAME OF YOUR DATA FILE FOLLOWED',
     1' BY <CR>.',/,' IF YOU DESIRE TO MAKE A TEST RUN USING THE',
     2' SAMPLE DATA ALREADY STORED',/,' IN THE FILE: CONCOR.TST',
     3'        TYPE: CONCOR.TST <CR>',/,' FILENAME: ',\,)
1119  format(a12)
1120  format(2x,a6,i4,5x,a6,i4,5x,'Error.')
2000  format(i1)
2001  format(i3)
2003  format(a18)
2004  format(a12)
2005  format(a5)
2010  format(e12.6,2x,e12.6)

      end


C                         SUBROUTINE: convol


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE ARRAYS xn1() and
C            xn2(), COMPUTES THE LINEAR CONVOLUTION OF THE ARRAYS,
C            AND RETURNS THE RESULTING SEQUENCE IN THE ARRAY yn().


      subroutine convol(ns1,N1,ns2,N2,ns3,ne3,xn1,xn2,yn)
      real xn1(-128:128), xn2(-128:128), yn(-256:256)

      j = 0

      do 10 n=ns3, ne3
     yn(n) = 0.0
     do 20 i=j, 0, -1
       if((j-i.lt.N2).and.(i.lt.N1)) then
         yn(n) = yn(n) + xn2(ns2+j-i)*xn1(ns1+i)
       endif
20       continue
     j = j + 1
10    continue

      return
      end
```

```
C                          SUBROUTINE: correl


C  PURPOSE:  THIS SUBROUTINE ACCEPTS AS INPUT THE ARRAYS xn1() AND
C            xn2(), AND COMPUTES THE LINEAR CORRELATION OF THE ARRAYS
C            BY THE FORMULA R(p) = SUM[xn1(n)*xn2(n+p)] FOR
C            ns3 <= p <= ne3.


      subroutine correl(ns1,ne1,ns2,ne2,ns3,ne3,xn1,xn2,R)
      real xn1(-128:128), xn2(-128:128), R(-256:256)
      integer p

      j = 0
      do 30 p=ns3, ne3
     R(p) = 0.0
     do 40 i=j, 0, -1
       index1 = ne1-j+i
       index2 = ns2+i
       if((index1.ge.ns1).and.(index2.le.ne2)) then
         R(p) = R(p) + xn1(index1)*xn2(index2)
       endif
40      continue
     j = j + 1
30    continue

      return
      end



C                          SUBROUTINE: sampl1


C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE SAMPLES
C            OF A CONTINUOUS FUNCTION AND STORE THEM IN THE ARRAY
C            xn1().  IF dsrce1 = 'S' THEN THE MAIN PROGRAM WILL CALL
C            THIS SUBROUTINE TO GENERATE THE VALUES FOR xn1().
C            IF dsrce1 DOES NOT EQUAL 'S' THEN THIS SUBROUTINE WILL
C            NOT BE CALLED BY THE MAIN PROGRAM.


      subroutine sampl1(ns1,ne1,xn1)
      real xn1(-128:128)

C***********************************************************************

C  DEVELOP THE SAMPLING ALGORITHM FOR xn1() IN THIS SPACE.  THE
C  STATEMENTS TYPED IN MUST FOLLOW STANDARD FORTRAN 77 RULES AND
C  MAY USE FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS: SIN(),COS()...
C  AN EXAMPLE OF AN ALGORITHM GENERATING VALUES FOR xn1() IS SHOWN.
C
```

```
C ***  EXAMPLE  ***
C
C      do 6 n=ns1, ne1
C       xn1(n) = 1.0
C 6    continue


C************************************************************************


      return
      end


C                         SUBROUTINE: sampl2


C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE SAMPLES
C            OF A CONTINUOUS FUNCTION AND STORE THEM IN THE ARRAY
C            xn2().  IF dsrce2 = 'S' THEN THE MAIN PROGRAM WILL CALL
C            THIS SUBROUTINE TO GENERATE THE VALUES FOR xn2().
C            IF dsrce2 DOES NOT EQUAL 'S' THEN THIS SUBROUTINE WILL
C            NOT BE CALLED BY THE MAIN PROGRAM.


      subroutine sampl2(ns2,ne2,xn2)
      real xn2(-128:128)

C************************************************************************

C  DEVELOP THE SAMPLING ALGORITHM FOR xn2() IN THIS SPACE.  THE
C  STATEMENTS TYPED IN MUST FOLLOW STANDARD FORTRAN 77 RULES AND
C  MAY USE FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS: SIN(),COS()...
C  AN EXAMPLE OF AN ALGORITHM GENERATING VALUES FOR xn2() IS SHOWN.
C
C ***  EXAMPLE  ***
C
C      do 7 n=ns2, ne2
C        xn2(n) = n + 1.0
C 7    continue

C************************************************************************


      return
      end
```

```
C                          DIFFEQ.FOR          VERSION: 2/03/88
C
C
C  PURPOSE:  THIS PROGRAM COMPUTES THE ITERATIVE SOLUTION TO A
C            LINEAR, TIME-INVARIANT (LTI) DIFFERENCE EQUATION.
C            THE DIFFERENCE EQUATION MUST BE IN THE FORM:
C            y(ns) = a(1)*y(ns-1) + ... + a(N)*y(ns-N) +
C                    b(0)*x(ns) + b(1)*x(ns-1) + ... + b(L)*x(ns-L).
C            THE PROGRAM CONSISTS OF A MAIN PROGRAM AND TWO
C            SUBROUTINES.  SUBROUTINE diffeq IS CALLED BY THE MAIN
C            PROGRAM TO ITERATIVELY SOLVE THE DIFFERENCE EQUATIONS AND
C            SUBROUTINE xgen ALLOWS THE USER THE OPTION OF GENERATING
C            THE INPUT SEQUENCE x() BY WRITING THE APPROPRIATE
C            EQUATIONS.  IF THE USER ELECTS TO GENERATE THE SEQUENCE
C            x() BY USING xgen THEN THE PROGRAM MUST BE COMPILED
C            AGAIN BEFORE EXECUTION.  THE USER HAS THE OPTION OF
C            SELECTING ONE OF TWO OPERATING MODES: BATCH OR TEST.
C            IN BATCH MODE THE AMOUNT OF INTERFACE WITH THE USER
C            IS MINIMIZED AND IT IS ASSUMED THAT THE INPUT DATA
C            HAS BEEN STORED IN THE DEFAULT FILE 'DIFFEQ.IN'.  IN
C            TEST MODE THE USER IS PROMPTED FOR THE NAME OF THE
C            INPUT FILE OR HAS THE OPTION OF PERFORMING A TEST RUN
C            USING THE INPUT DATA STORED IN THE FILE 'DIFFEQ.TST'.
C            IT IS RECOMMENDED THAT FIRST-TIME USERS SELECT THE
C            TEST MODE AND MAKE A TRIAL RUN WITH THE PRESTORED
C            INPUT DATA.  THE TEST MODE ECHOES PORTIONS OF THE
C            INPUT DATA ONTO THE MONITOR TO ALLOW VERIFICATION OF
C            ITS ACCURACY.  BOTH BATCH AND TEST MODES ALLOW THE
C            USER TO SOLVE UP TO FOUR DIFFERENCE EQUATIONS IN A
C            SINGLE PROGRAM EXECUTION.  THE OUTPUT OF THE PROGRAM
C            'DIFFEQ.FOR' IS STORED IN THE ARRAY y().  THE OUTPUT IS
C            STORED IN TABULAR FORM IN THE OUTPUT FILE 'DIFFEQ.FOR'
C            AND IN A FORM SUITABLE FOR PLOTTING IN THE FILE
C            'DIFFEQ.DAT'.
C
C
C******************************   INPUT   ********************************

C
C
C  THIS PROGRAM ASSUMES THAT EACH DIFFERENCE EQUATION IS IN THE
C  CANONICAL FORM:
C
C  y(ns) = a(1)*y(ns-1) + ... + a(N)*y(ns-N) +
C          b(0)*x(ns) + b(1)*x(ns-1) + ... + b(L)*x(ns-L)
C
C  L = A NON-NEGATIVE INTEGER, THE NUMBER OF INPUT DELAYS.
C  N = A NON-NEGATIVE INTEGER, THE NUMBER OF OUTPUT DELAYS.
C
C  a(1)...a(N) = REAL COEFFICIENTS OF THE OUTPUT TERMS.
C  b(0)...b(L) = REAL COEFFICIENTS OF THE INPUT TERMS.
C
```

```
C  THE INPUT PARAMETERS SHOULD BE STORED IN A FILE NAMED
C  'DIFFEQ.IN'.  ALL OF THE READ STATEMENTS USED BY THIS PROGRAM
C  REQUIRE FORMATTED INPUT.  PARTICULAR ATTENTION SHOULD BE PAID
C  TO THE FORMATS, ESPECIALLY THE USE OF THE DECIMAL POINT TO
C  DENOTE 'REAL' NUMBERS.  THE INPUT PARAMETERS REQUIRED BY THE
C  PROGRAM ARE LISTED BELOW.
C
C
C  NAME         TYPE       RANGE (ARRAYS)         RESTRICTIONS
C  ----         ----       --------------       ----------------
C  numsys       INTEGER                          1 <= numsys <= 4
C  L            INTEGER                            0 <= L <= 128
C  N            INTEGER                            0 <= N <= 128
C  nstop        INTEGER                          0 <= nstop <= 300
C  xsorce       CHARACTER                            'F' OR 'S'
C  b(k)         REAL         0,1, ..., L           0 <= L <= 128
C  a(k)         REAL         1,2, ..., N           0 <= N <= 128
C  y(k)         REAL         -N, ..., -1           1 <= N <= 128
C  x(ns)        REAL       0,1, ..., nstop       0 <= nstop <= 300
C
C  WHERE:
C
C  numsys = THE NUMBER OF SYSTEMS TO BE EVALUATED.
C           THIS INTEGER VALUE MUST OCCUR AT THE TOP OF THE INPUT
C           FILE. IT DELINEATES THE NUMBER OF SYSTEMS TO BE READ BY
C           THE PROGRAM AND ANALYZED.  FOR EACH SYSTEM (1...numsys)
C           THE PARAMETERS BELOW MUST APPEAR IN THE INPUT FILE.
C
C  L = AN INTEGER VALUE THAT SPECIFIES THE MAXIMUM NUMBER OF DELAYS
C      IN THE INPUT SEQUENCE.
C
C  N = AN INTEGER VALUE THAT SPECIFIES THE MAXIMUM NUMBER OF DELAYS
C      IN THE OUTPUT SEQUENCE.
C
C  nstop = AN INTEGER VALUE THAT SPECIFIES THE LARGEST TIME INDEX
C          (ns) FOR WHICH THE DIFFERENCE EQUATION IS TO BE SOLVED.
C
C  xsorce = A CHARACTER VALUE OF 'F' OR 'S' DENOTING WHETHER THE
C           INPUT SEQUENCE x() IS TO BE READ FROM THE INPUT FILE (F)
C           OR TO BE GENERATED (S) USING THE SUBROUTINE xgen.  THIS
C           LATER OPTION IS ATTRACTIVE WHEN nstop IS A LARGE NUMBER
C           AND THE INPUT SEQUENCE x() CAN BE READILY DESCRIBED BY AN
C           ANALYTICAL EXPRESSION.  IF xsorce = 'S' THE USER MUST
C           PROVIDE THE APPROPRIATE FORTRAN STATEMENTS IN THE SPACE
C           PROVIDED IN SUBROUTINE xgen AND THE PROGRAM MUST BE
C           RECOMPILED BEFORE EXECUTION.
C
C  b(k)  = REAL COEFFICIENTS OF THE INPUT SEQUENCE x(ns-k) IN THE
C         ORDER:  b(0), b(1), ..., b(L).
C
```

```
C  a(k) = REAL COEFFICIENTS OF THE OUTPUT SEQUENCE y(ns-k) IN THE
C         ORDER:  a(1), a(2), ..., a(N).  IF N = 0 THEN THE EQUATION
C         IS NON-RECURSIVE AND NO a(k) COEFFICIENTS SHOULD BE IN
C         THE INPUT FILE.
C
C  y(k) = THE INITIAL CONDITIONS FOR THE OUTPUT SEQUENCE IF THE
C         DIFFERENCE EQUATION IS RECURSIVE, I.E., N > 0.  THIS
C         PROGRAM CALCULATES THE SOLUTION TO THE DIFFERENCE
C         EQUATION FROM ns = 0 TO ns = nstop THEREFORE THE INITIAL
C         CONDITIONS y(-N) TO y(-1) MUST BE PROVIDED IN THE INPUT
C         FILE IN THE ORDER: y(-N), y(-N+1), ..., y(-1).  IF N = 0
C         THEN THE EQUATION IS NON-RECURSIVE AND NO INITIAL
C         CONDITIONS SHOULD BE GIVEN IN THE INPUT FILE.
C
C  x(ns) = THE INPUT SEQUENCE.  IF xsorce = 'F' THEN THE INPUT
C          SEQUENCE x(0), ..., x(nstop) MUST BE PROVIDED BY THE USER
C          IN THE INPUT FILE.  IF xsorce = 'S' THEN THE USER HAS
C          ELECTED TO GENERATE THE INPUT SEQUENCE BY PROVIDING THE
C          APPROPRIATE FORTRAN STATEMENTS IN THE SUBROUTINE xgen.
C
C  NOTE:  THE INPUT FORMAT STATEMENTS OCCUR IN THE MAIN PROGRAM
C         FOLLOWING THE CAPTION: ********  INPUT FORMAT  ********.
C         THE FORM OF THE INPUT DATA FILE IS:
C
C  LINE #                      ENTRIES                    FORMAT
C  ______                      _______                    ______
C    1                         numsys                       i1
C    2                    L,N,nstop,xsorce        i3,t11,i3,t21,i3,t31,a1
C  NEXT NB LINES          b(k), k=0,1,...,L               6f10.0
C  NEXT NA LINES           a(k), k=1,...,N                6f10.0
C  NEXT NY LINES          y(k), k= -N,...,-1              6f10.0
C  NEXT NX LINES       x(ns), ns= 0,...,nstop             6f10.0
C
C  WHERE: NB = 1 + (L/6 ROUNDED DOWN TO THE NEXT SMALLER INTEGER)
C
C         NA = 0 IF N = 0 OR
C         NA = 1 + ((N-1)/6 ROUNDED DOWN TO THE NEXT SMALLER INTEGER)
C
C         NY = 0 IF N = 0 OR
C         NY = 1 + ((N-1)/6 ROUNDED DOWN TO THE NEXT SMALLER INTEGER)
C
C         NX = 0 IF xsorce = 'S' OR
C         NX = 1 + (nstop/6 ROUNDED DOWN TO THE NEXT SMALLER INTEGER)
C              IF xsorce = 'F'
C
C  *NOTE:  FOR numsys > 1 THE FORMAT OF LINES 2... IS REPEATED.
C
C          THE FORMAT f10.0 USED FOR INPUT DATA PERMITS THE DECIMAL
C          POINT TO BE PLACED ANYWHERE IN THE FIELD OF 10 COLUMNS
C          AND ALSO ALLOWS THE EXPONENTIAL FORMAT TO BE USED (EG.
C          3146.2 = 3.1462E+03).
C
```

```
C
C******************************  OUTPUT   ******************************
C
C
C  THE INPUT DATA AS WELL AS THE OUTPUT DATA ARE STORED IN TABULAR
C  FORM IN THE FILE 'DIFFEQ.OUT'.  ADDITIONALLY, THE INPUT SEQUENCE
C  AND THE OUTPUT SEQUENCE ARE WRITTEN INTO THE FILE 'DIFFEQ.DAT' TO
C  FACILITATE PLOTTING BY A SEPARATE, USER SUPPLIED PROGRAM.  THE
C  FORMAT OF THE DATA IN 'DIFFEQ.DAT' IS: e12.6, 2X, e12.6. THE FIRST
C  ENTRY CORRESPONDS TO THE ORDINATE VALUE, AND THE SECOND ENTRY, THE
C  ABSCISSA VALUE.  ADDITIONAL HEADER INFORMATION IS WRITTEN INTO
C  'DIFFEQ.DAT' TO ALLOW FOR CONTROL AND LABELING OF EACH PLOT.
C
C
C******************************  EXAMPLE  ******************************
C
C
C  THE INPUT PARAMETERS FOR THE SYSTEM DESCRIBED BELOW ARE STORED IN
C  THE SAMPLE INPUT FILE 'DIFFEQ.TST' AND CAN BE USED FOR A TRIAL
C  RUN IN TEST MODE.
C
C  DIFFERENCE EQUATION:
C
C  y(ns) = 1.2*y(ns-1) + 1.5*x(ns)
C
C  GOAL:    TO OBTAIN THE SOLUTION TO THIS DIFFERENCE EQUATION FOR
C           ns = 0 TO ns = 10, GIVEN: x(0)...x(10) = 100.0 AND
C           THE INITIAL CONDITION y(-1) = 25.0.
C
C
C  THE INPUT FILE IS:
C
C  1
C  000       001       010       F
C  1.5
C  1.2
C  25.0
C  100.0     100.0     100.0     100.0     100.0     100.0
C  100.0     100.0     100.0     100.0     100.0
C
C
C  THE RESULTING OUTPUT FILE 'DIFFEQ.OUT' IS:
C
C                INPUT DATA FOR PROBLEM #  1
C
C   PROBLEM # 1    INPUT DATA SOURCEFILE: DIFFEQ.TST
C   THE NUMBER OF INPUT DELAYS: L =   0
C   THE NUMBER OF OUTPUT DELAYS: N =   1
C   THE VALUE OF nstop IS:  10
```

```
C   THE COEFFICIENTS b(0), b(1), ..., b(L) ARE:
C
C  .150000E+01
C
C
C   THE COEFFICIENTS a(1), ..., a(N) ARE:
C
C  .120000E+01
C
C
C                 OUTPUT DATA FOR PROBLEM #  1
C
C     ns         x(ns)               y(ns)
C     -1      .000000E+00        .250000E+02
C      0      .100000E+03        .180000E+03
C      1      .100000E+03        .366000E+03
C      2      .100000E+03        .589200E+03
C      3      .100000E+03        .857040E+03
C      4      .100000E+03        .117845E+04
C      5      .100000E+03        .156414E+04
C      6      .100000E+03        .202697E+04
C      7      .100000E+03        .258236E+04
C      8      .100000E+03        .324883E+04
C      9      .100000E+03        .404860E+04
C     10      .100000E+03        .500832E+04
C
C --------------- END OF PROBLEM # 1  ---------------
C
C
C  FOR ILLUSTRATIVE PURPOSES THE INPUT SEQUENCE x() COULD HAVE BEEN
C  GENERATED BY SPECIFYING xsorce = 'S' AND WRITING THE APPROPRIATE
C  FORTRAN STATEMENTS INTO SUBROUTINE xgen.  THE STATEMENTS THAT
C  COULD BE USED TO ACCOMPLISH THIS ARE WRITTEN INTO THE SUBROUTINE
C  BUT ARE 'COMMENTED OUT'.
C
C
C***************************  MAIN PROGRAM  ***************************



      character infile*12, mode*1, xsorce*1
      real a(1:128), b(0:128), y(-128:300), x(-128:300), ii

C  PROMPT USER FOR MODE: BATCH OR TEST.

      write(*,1115)
      read(*,1117) mode
      if((mode.eq.'Y').or.(mode.eq.'y')) then
     mode = 'Y'
     write(*,1118)
     read(*,1119) infile
      else
     infile = 'DIFFEQ.IN'
      endif

C  UNIT=1 DEFINED AS INPUT FILE.  UNITS=2,3 DEFINED AS OUTPUT FILES.

      open(unit=1,file=infile,status='old',iostat=ierr,err=999)
      open(unit=2,file='DIFFEQ.OUT')
      open(unit=3,file='DIFFEQ.DAT')

C  READ INPUT PARAMETERS AND CONDUCT ERROR CHECKS.

      read(1,1000) numsys

      if((numsys.le.0).or.(numsys.gt.4)) then
     write(*,1126) numsys
     stop 'The allowed values for numsys are 1 <= numsys <= 4.'
      endif
      numplts = 2*numsys
      write(3,2000) numplts

      do 10 nprob=1, numsys

     read(1,1001) L, N, nstop, xsorce

     if((L.lt.0).or.(L.gt.128)) then
       write(*,1124) nprob, 'L', L
       stop 'The allowed values for ''L'' are:  0 <= L <= 128.'
     endif

     if((N.lt.0).or.(N.gt.128)) then
       write(*,1124) nprob, 'N', N
       stop 'The allowed values for ''N'' are:  0 <= N <= 128.'
     endif

     if((nstop.lt.0).or.(nstop.gt.300)) then
       write(*,1127) nprob, nstop
       stop 'The allowed values for nstop are: 0 <= nstop <= 300.'
     endif

     if((xsorce.eq.'F').or.(xsorce.eq.'f')) then
       xsorce = 'F'
     elseif((xsorce.eq.'S').or.(xsorce.eq.'s')) then
       xsorce = 'S'
     else
       write(*,1128) nprob, xsorce
       stop 'The allowed values for ''xsorce'' are: ''F'' or ''S''.'
     endif

C  INITIALIZE EACH ARRAY TO ZERO BEFORE EACH RUN.

     data a/128*0.0/, b/129*0.0/
     data y/429*0.0/, x/429*0.0/
```

```
C  READ THE COEFFICIENTS b(), a() AND THE INITIAL CONDITIONS
C  y(-N)...y(-1).

      read(1,1002) (b(k), k=0,L)
      if(N.gt.0) then
        icstart = -N
        read(1,1002) (a(k), k=1,N)
        read(1,1002) (y(k), k=icstart,-1)
      endif

C  FOR xsorce = 'F' READ THE ARRAY x() FROM THE INPUT FILE.
C  FOR xsorce = 'S' CALL xgen TO GENERATE THE ARRAY x().

      if(xsorce.eq.'F') then
        read(1,1002) (x(k), k=0,nstop)
      else
        call xgen(x,nstop,nprob)
      endif

C  FOR TEST MODE ECHO INPUT PARAMETERS ONTO MONITOR (UNIT = *).

      if(mode.eq.'Y') then
        write(*,1007)
        write(*,1120) nprob, infile
        write(*,1110) 'INPUT', 'L', L
        write(*,1110) 'OUTPUT', 'N', N
        write(*,1112) nstop
        write(*,1004)
        write(*,1005) (b(k),k=0,L)
        if(N.eq.0) then
          write(*,1131)
        else
          write(*,1006)
          write(*,1005) (a(k),k=1,N)
        endif
        write(*,1123) nprob
        pause 'END OF RUN, STRIKE <CR> WHEN READY TO CONTINUE.'
      endif

C  WRITE INPUT DATA INTO FILE: DIFFEQ.OUT.

      write(2,1008) 'INPUT', nprob
      write(2,1120) nprob, infile
      write(2,1110) 'INPUT', 'L', L
      write(2,1110) 'OUTPUT', 'N', N
      write(2,1112) nstop
      write(2,1004)
      write(2,1005) (b(k),k=0,L)
```

```
     if(N.eq.0) then
       write(2,1131)
     else
       write(2,1006)
       write(2,1005) (a(k),k=1,N)
     endif

C  WRITE THE INPUT SEQUENCE INTO FILE: DIFFEQ.DAT.

     write(3,2001) nstop + 1
     write(3,*) 'INPUT SEQUENCE x(ns)'
     write(3,*) 'SAMPLE # (ns)'
     write(3,*) 'x(ns)'
     do 54 ns=0, nstop
       ii = ns
       write(3,2010) ii, x(ns)
54      continue

C  CALL diffeq TO COMPUTE THE SOLUTION TO THE DIFFERENCE EQUATION.

     call diffeq(N,L,a,b,x,y,nstop)

C  WRITE RESULTS INTO FILE: DIFFEQ.DAT.

     write(3,2001) N + nstop + 1
     write(3,*) 'OUTPUT SEQUENCE Y(ns)'
     write(3,*) 'SAMPLE # (ns)'
     write(3,*) 'y(ns)'
     do 55 ns= -N, nstop
       ii = ns
       write(3,2010) ii, y(ns)
55      continue

C  WRITE RESULTS INTO FILE: DIFFEQ.OUT.

     write(2,1008) 'OUTPUT', nprob
     write(2,1129)
     do 102 ns= -N, nstop
       write(2,1130) ns, x(ns), y(ns)
102     continue
     write(2,1123) nprob

10    continue

      write(*,1121)
999   close(unit=1)
      close(unit=2)
      close(unit=3)

      if(ierr.gt.0) then
     write(*,1116) ierr
      endif
```

```
C********  INPUT FORMAT  ********

1000  format(i1)
1001  format(i3,t11,i3,t21,i3,t31,a1)
1002  format(6f10.0)

C********************************

1004  format(t4,'THE COEFFICIENTS b(0), b(1), ..., b(L) ARE:',/)
1005  format(6(1x,e12.6))
1006  format(//,t4,'THE COEFFICIENTS a(1), ..., a(N) ARE:',/)
1007  format(//////////)
1008  format(///,t16,a6,' DATA FOR PROBLEM # ',i1,//)
1110  format(t4,'THE NUMBER OF ',a6,' DELAYS: ',a1,' = ',i3)
1112  format(t4,'THE VALUE OF nstop IS: ',i3)
1115 0format(1x,'DO YOU WISH TO RUN THIS PROGRAM IN TEST',
     1' MODE ?    (Y/N) <CR> : ',\,)
1116 0format(///,1x,'ERROR OPENING INPUT FILE, PROGRAM TERMINATED.',
     1//,1x,'ERROR CODE:',i4,/////)
1117  format(a1)
1118 0format(/////,1x,'TYPE THE NAME OF YOUR DATA FILE FOLLOWED',
     1' BY <CR>.',/,' IF YOU DESIRE TO MAKE A TEST RUN USING THE',
     2' SAMPLE DATA ALREADY STORED',/,' IN THE FILE: DIFFEQ.TST',
     3 '        TYPE: DIFFEQ.TST <CR>',/,' FILENAME: ',\,)
1119  format(a12)
1120  format(///,t4,'PROBLEM # ',i1,'   INPUT DATA SOURCEFILE: ',a12)
1121 0format(//,' TABULAR OUTPUT DATA IS STORED IN FILE: DIFFEQ.OUT.'
     1,/,' PLOTTING DATA IS STORED IN FILE: DIFFEQ.DAT.')
1122  format(i3)
1123  format(/,1x,16('-'),' END OF PROBLEM #',i2,2x,16('-'),//)
1124 0format(//,' For problem #',i2,' the value for ',a1,' is: ',i3,
     1'.  This value is not allowed.')
1126  format(//,' numsys = ',i4,'.  This value is not allowed.')
1127 0format(//,' For problem #',i2,' the value for ''nstop'' is: ',
     1i3,'.  This value is not allowed.')
1128 0format(//,' For problem #',i2,' the value for ''xsorce'' is: ',
     1a1,'.  This value is not allowed.')
1129  format(t6,'ns',t16,'x(ns)',t35,'y(ns)')
1130  format(t4,i4,t11,e14.6,t30,e14.6)
1131  format(/,' THIS SYSTEM IS NON-RECURSIVE, I.E., N = 0.')
2000  format(i1)
2001  format(i3)
2010  format(e12.6,2x,e12.6)

      end
```

```
C                              SUBROUTINE: diffeq


C  PURPOSE:  THIS SUBROUTINE COMPUTES THE SOLUTION TO A DIFFERENCE
C            EQUATION.  ALL PARAMETERS DESCRIBING THE EQUATION, AND
C            THE INPUT AND OUTPUT SEQUENCES x() AND y() ARE PASSED
C            TO THE SUBROUTINE BY THE MAIN PROGRAM.

      subroutine diffeq(N,L,a,b,x,y,nstop)
      real x(-128:nstop), y(-128:nstop), a(1:N), b(0:L)

      do 500 ns=0, nstop
     y(ns) = 0.0
     do 501 k=0, max(N,L)
       y(ns) = y(ns) + a(k)*y(ns-k) + b(k)*x(ns-k)
501      continue
500   continue

      return
      end


C                              SUBROUTINE: xgen


C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE VALUES FOR
C            THE ARRAY x().  IF xsorce = 'S' THE MAIN PROGRAM WILL
C            CALL THIS SUBROUTINE.  IF xsorce = 'F' THIS SUBROUTINE
C            WILL NOT BE CALLED BY THE MAIN PROGRAM.


      subroutine xgen(x,nstop,nprob)
      real x(-128:nstop)

C**********************************************************************

C  DEVELOP THE ALGORITHM FOR GENERATING VALUES OF x() IN THIS SPACE.
C  THE STATEMENTS TYPED IN MUST FOLLOW STANDARD FORTRAN 77 RULES AND
C  MAY USE FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS: SIN(), COS(), ...
C  NOTE THAT THE VALUE nprob CAN BE USED IN A LOGICAL 'IF' STATEMENT
C  TO MATCH THE GENERATING FUNCTIONS TO THE CORRESPONDING SYSTEM
C  EQUATION READ FROM THE INPUT FILE IF MORE THAN ONE SYSTEM OF
C  EQUATIONS EXIST.  AN EXAMPLE OF AN ALGORITHM GENERATING VALUES
C  FOR x() IS:
C
C
```

```
C***  EXAMPLE  ***
C
C     if(nprob.eq.1) then
C       do 1 k=0, nstop
C         x(k) = 100.0
C 1     continue
C     endif

C*************************************************************************


      return
      end
```

```
C                          STATEQ.FOR        VERSION: 2/03/88
C
C
C  PURPOSE:  THIS PROGRAM COMPUTES THE ITERATIVE SOLUTION TO A SET
C            OF LINEAR, TIME-INVARIANT STATE EQUATIONS.  THE FORM
C            OF THE STATE EQUATIONS IS:
C
C            v(ns+1) = Av(ns) + Bx(ns)
C            y(ns) = Cv(ns) + Dx(ns)
C
C            WHERE A, B, C, D ARE MATRICES OF THE SYSTEM CONSTANTS,
C            v IS THE STATE VECTOR, x IS THE INPUT VECTOR, AND y
C            IS THE OUTPUT VECTOR.  THE PROGRAM CONSISTS OF A MAIN
C            PROGRAM AND TWO SUBROUTINES.  THE SUBROUTINE xgen GIVES
C            THE USER THE OPTION OF GENERATING THE VECTOR INPUT
C            SEQUENCE x() BY WRITING THE APPROPRIATE EQUATIONS IN THE
C            SPACE ALLOCATED IN THIS SUBROUTINE.  IF THE USER CHOOSES
C            TO GENERATE THE INPUT DATA BY USING SUBROUTINE xgen THE
C            EQUATIONS MUST BE WRITTEN INTO THE SUBROUTINE USING
C            STANDARD FORTRAN 77 EXECUTABLE STATEMENTS AND THE VALUES
C            GENERATED MUST BE STORED IN THE 2-DIMENSIONAL ARRAY
C            xs().   THE SUBROUTINE itrate COMPUTES THE STATE OF THE
C            SYSTEM v(), AS WELL AS THE OUTPUT OF THE SYSTEM y(), FOR
C            EACH VALUE OF ns FROM ns = 0 TO ns = nstop.  THE USER
C            HAS THE OPTION OF SELECTING ONE OF TWO OPERATING MODES:
C            BATCH OR TEST.  IN BATCH MODE THE AMOUNT OF INTERFACE
C            WITH THE USER IS MINIMIZED AND IT IS ASSUMED THAT THE
C            INPUT PARAMETERS DESCRIBED BELOW HAVE BEEN STORED IN
C            THE INPUT FILE 'STATEQ.IN'.  IN TEST MODE THE USER IS
C            PROMPTED FOR THE NAME OF THE INPUT FILE AND HAS THE
C            OPTION OF PERFORMING A TEST RUN USING THE DATA STORED IN
C            IN THE FILE 'STATEQ.TST'.  IT IS RECOMMENDED THAT FIRST-
C            TIME USERS SELECT TEST MODE AND PERFORM A TRIAL RUN
C            WITH THE PRESTORED DATA. THE TEST MODE ECHOES PORTIONS
C            OF THE INPUT DATA ONTO THE MONITOR TO ALLOW VERIFICATION
C            OF ITS ACCURACY.  THE OUTPUT OF THE PROGRAM 'STATEQ.FOR'
C            IS STORED IN THE 2-DIMENSIONAL ARRAY ys() AND THE SYSTEM
C            STATES ARE STORED IN THE 2-DIMENSIONAL ARRAY vs().  THE
C            INPUT VALUES, SYSTEM STATES AND THE OUTPUT DATA ARE
C            STORED IN TABULAR FORM IN THE FILE 'STATEQ.OUT' AND IN A
C            FORM SUITABLE FOR PLOTTING IN THE FILE 'STATEQ.DAT'.
C
C
C******************************   INPUT   ********************************

C
C
C  THIS PROGRAM ASSUMES THAT THE STATE EQUATIONS ARE IN THE
C  FORM :
C
C  [v1(ns+1)]   [ a11 ... aNN] [ v1(ns)  ]   [b11 ... b1M] [x1(ns)]
C  [v2(ns+1)]   [ a21 ... a2N] [ v2(ns)  ]   [b21 ... b2M] [x2(ns)]
```

```
C  [   ...    ] = [ ... ... ...]*[  ...    ] + [... ... ...]*[ ...  ]
C  [vN(ns+1)]    [ aN1 ... aNN] [ vN(ns) ]    [bN1 ... bNM] [xM(ns)]
C
C
C
C  [y1(ns)]    [c11 ... c1N] [v1(ns)]    [d11 ... d1M] [x1(ns)]
C  [y2(ns)]    [c21 ... c2N] [v2(ns)]    [d21 ... d2M] [x2(ns)]
C  [  ...  ] = [... ... ...]*[  ...  ] + [... ... ...]*[ ...  ]
C  [yQ(ns)]    [cQ1 ... cQN] [vN(ns)]    [dQ1 ... dQM] [xM(ns)]
C
C  N IS THE NUMBER OF SYSTEM STATES.
C  M IS THE NUMBER OF SYSTEM INPUTS.
C  Q IS THE NUMBER OF SYSTEM OUTPUTS.
C
C  x IS THE M x 1 INPUT VECTOR.
C  v IS THE N x 1 STATE VECTOR.
C  y IS THE Q x 1 OUTPUT VECTOR.
C  A IS AN N x N MATRIX OF CONSTANTS.
C  B IS AN N x M MATRIX OF CONSTANTS.
C  C IS A Q x N MATRIX OF CONSTANTS.
C  D IS A Q x M MATRIX OF CONSTANTS.
C
C
C  THE SOLUTION IS GENERATED IN THE INTERVAL 0 <= ns <= nstop.  THE
C  USER MUST PROVIDE THE MATRICES A, B, C, D AS WELL AS THE INITIAL
C  VALUES OF THE STATE VECTOR v IN THE INPUT FILE. THESE INPUTS, AS
C  WELL AS THE PARAMETERS DESCRIBED BELOW, SHOULD BE STORED IN THE
C  INPUT FILE 'STATEQ.IN'.  ALL OF THE READ STATEMENTS USED BY THIS
C  PROGRAM REQUIRE FORMATTED INPUT.  PARTICULAR ATTENTION SHOULD BE
C  PAID TO THESE FORMATS, ESPECIALLY THE USE OF THE DECIMAL POINT
C  TO DISTINGUISH BETWEEN 'REAL' AND INTEGER DATA.
C
C
C
C  NAME         TYPE       RANGE (ARRAYS)        RESTRICTIONS
C  ----         ----       --------------        ------------
C  N           INTEGER                            0 <= N <= 10
C  M           INTEGER                            0 <= M <= 4
C  Q           INTEGER                            0 <= Q <= 4
C  nstop       INTEGER                          0 <= nstop <= 99
C  xsorce      CHARACTER                           'F' OR 'S'
C  A(i,j)      REAL         i=1,...,N             0 <= N <= 10
C                           j=1,...,N
C  B(i,j)      REAL         i=1,...,N             0 <= N <= 10
C                           j=1,...,M             0 <= M <= 4
C  C(i,j)      REAL         i=1,...,Q             0 <= Q <= 4
C                           j=1,...,N             0 <= N <= 10
C  D(i,j)      REAL         i=1,...,Q             0 <= Q <= 4
C                           j=1,...,M             0 <= M <= 4
C  v(i)        REAL         i=1,...,N             0 <= N <= 10
C  xs(i,j)     REAL         i=1,...,M             0 <= M <= 4
C                           j=0,...,nstop       0 <= nstop <= 99
```

```
C
C
C  WHERE:
C
C  N = AN INTEGER VALUE THAT SPECIFIES THE NUMBER OF SYSTEM STATES,
C      I.E., THE ORDER OF THE SYSTEM.
C
C  M = AN INTEGER VALUE THAT SPECIFIES THE NUMBER OF SYSTEM INPUTS.
C
C  Q = AN INTEGER VALUE THAT SPECIFIES THE NUMBER OF SYSTEM OUTPUTS.
C
C  nstop = AN INTEGER VALUE THAT SPECIFIES THE LARGEST TIME INDEX
C          (ns) FOR WHICH THE STATE EQUATIONS ARE TO BE SOLVED.
C
C  xsorce = A CHARACTER VALUE OF 'F' OR 'S' DENOTING WHETHER THE INPUT
C           SEQUENCE(S) xs(i,j) ARE TO BE READ FROM THE INPUT FILE (F)
C           OR TO BE GENERATED (S) USING THE SUBROUTINE xgen.  THIS
C           LATTER OPTION IS ATTRACTIVE WHEN nstop IS A LARGE NUMBER
C           AND THE INPUT SEQUENCE(S) xs(i,j) CAN BE READILY DESCRIBED
C           BY ANALYTICAL EXPRESSIONS.  IF xsorce = 'F' THE VALUES OF
C           xs(i,j), i=1,...,M ; j=0,...,nstop  ARE READ FROM THE
C           INPUT FILE.  IF xsorce = 'S' THEN THE USER HAS ELECTED
C           TO GENERATE THE INPUT SEQUENCE(S) xs(1,j),...,xs(M,j) BY
C           WRITING THE APPROPRIATE FORTRAN STATEMENTS IN THE SUB-
C           ROUTINE xgen.  IF THIS METHOD OF INPUT DATA GENERATION
C           IS SELECTED NO VALUES OF THE INPUT SEQUENCE SHOULD APPEAR
C           IN THE INPUT FILE AND THE PROGRAM MUST BE RECOMPILED
C           BEFORE EXECUTION.
C
C  A(i,j) = AN N x N MATRIX OF REAL NUMBERS.
C
C  B(i,j) = AN N x M MATRIX OF REAL NUMBERS.
C
C  C(i,j) = A Q x N MATRIX OF REAL NUMBERS.
C
C  D(i,j) = A Q x M MATRIX OF REAL NUMBERS.
C
C  v(i) = THE N x 1 INITIAL CONDITION VECTOR OF THE SYSTEM STATES.
C         THE USER MUST PROVIDE THE VALUES OF THE STATES FOR ns=0,
C         I.E.,  v(1),...,v(N).  THESE VALUES ARE THE INITIAL
C         CONDITIONS OF THE SYSTEM.
C
C  xs(i,j) = AN M x (nstop+1) MATRIX OF REAL NUMBERS. THE SEQUENCE(S)
C            xs(1,ns),...,xs(M,ns) ARE THE INPUTS TO THE SYSTEM AT
C            SAMPLE ns.
C
C  NOTE:  THE INPUT FORMAT STATEMENTS OCCUR IN THE MAIN PROGRAM
C         FOLLOWING THE CAPTION: ********  INPUT FORMAT  ********.
C         THE FORM OF THE INPUT DATA FILE IS:
C
```

```
C  LINE #                  ENTRIES               FORMAT          NOTES
C  ------                  -------               ------          -----
C    1                      N,M,Q          i2,t11,i1,t21,i1
C    2                  nstop,xsorce          i2,t11,a1
C  NEXT NA LINES      A(i,j),i=1,...,N       6f10.0          READ BY ROWS
C                            j=1,...,N
C  NEXT N LINES       B(i,j),i=1,...,N       4f10.0          READ BY ROWS
C                            j=1,...,M
C  NEXT NC LINES      C(i,j),i=1,...,Q       6f10.0          READ BY ROWS
C                            j=1,...,N
C  NEXT Q LINES       D(i,j),i=1,...,Q       4f10.0          READ BY ROWS
C                            j=1,...,M
C  NEXT Nv LINES      v(i)   i=1,...,N       6f10.0
C  NEXT Nx LINES      xs(i,j),i=1,...,M      4f10.0          EACH LINE
C                             j=0,...,ntop                   CORRESPONDS
C                                                            TO ONE VALUE
C                                                            OF ns.
C
C  WHERE:
C
C   NA = N     IF  N <= 6
C        2*N   OTHERWISE.
C
C   NC = Q     IF N <= 6
C        2*Q   OTHERWISE.
C
C   Nv = 0     IF N = 0,
C   Nv = 1     IF 1 <= N <= 6,
C   Nv = 2     IF N > 6.
C
C   Nx = 0     IF  xsorce = 'S'  OR
C   Nx = nstop + 1   IF  xsorce = 'F'
C
C  NOTE:  THE FORMAT f10.0 USED FOR INPUT DATA PERMITS THE DECIMAL
C         POINT TO BE PLACED ANYWHERE IN THE FIELD OF 10 COLUMNS
C         AND ALSO ALLOWS THE EXPONENTIAL FORMAT TO BE USED (EG.
C         3146.2 = 3.1462E+03).
C
C
C*******************************  OUTPUT   ******************************

C
C
C  THE INPUT DATA, SYSTEM STATES, AND THE OUTPUT SEQUENCE(S) ARE
C  STORED IN TABULAR FORM IN THE FILE 'STATEQ.OUT'.  ADDITIONALLY,
C  UP TO 9 SEQUENCES CONSISTING OF THE INPUT AND OUTPUT SEQUENCES
C  AND THE SYSTEM STATES ARE STORED IN THE FILE 'STATEQ.DAT' TO
C  FACILITATE PLOTTING BY A SEPARATE, USER PROVIDED PROGRAM.  NOTE
C  THAT A MAXIMUM OF 9 SEQUENCES ARE WRITTEN INTO THIS FILE.  IF
C  MORE THAN 9 SEQUENCES EXIST, ALL OF THE INPUT AND OUTPUT SEQUENCES
C  (xs() AND ys()) WILL BE STORED, HOWEVER SOME OF THE SYSTEM STATES
C  (vs()) WILL NOT BE STORED IN 'STATEQ.DAT'.  THE FORMAT OF THE DATA
```

```
C  IN 'STATEQ.DAT' IS:  e12.6, 2X, e12.6. THE FIRST ENTRY CORRESPONDS
C  TO THE ORDINATE VALUE, AND THE SECOND ENTRY, THE ABSCISSA VALUE.
C  ADDITIONAL HEADER INFORMATION IS WRITTEN INTO 'STATEQ.DAT' TO
C  ALLOW FOR CONTROL AND LABELING OF EACH PLOT.
C
C
C******************************   EXAMPLE   ******************************

C
C
C  THE INPUT PARAMETERS FOR THE SYSTEM DESCRIBED BELOW ARE STORED
C  IN THE FILE 'STATEQ.TST'.  THE GOAL IS TO READ THE INPUT VALUES
C  FROM THE INPUT FILE AND TO CALCULATE THE STATE VECTOR v AND THE
C  CORRESPONDING OUTPUT VECTOR y FOR ns = 0 TO 3.
C
C  GIVEN: N = 2
C         M = 1
C         Q = 1
C
C   x1(ns) = 10.0*u(ns)
C   xsorce = 'F'  I.E., THE SEQUENCE xs(1,ns) IS READ
C            FROM THE INPUT FILE.
C
C   INITIAL CONDITIONS:
C
C   v1(0) =  5.0
C   v2(0) = -5.0
C
C  SYSTEM OF EQUATIONS IN MATRIX FORM:
C
C  [v1(ns+1)] = [0.0  -1.0]*[v1(ns)] + [1.0]*[x1(ns)]
C  [v2(ns+1)]   [1.0   0.0] [v2(ns)]   [0.0]
C
C  [y1(ns)]   = [1.0  -1.0]*[v1(ns)] + [1.0]*[x1(ns)]
C                           [v2(ns)]
C
C  THE INPUT FILE IS:
C
C  02        1         1
C  03        F
C  0.0       -1.0
C  1.0       0.0
C  1.0
C  0.0
C  1.0       -1.0
C  1.0
C  5.0       -5.0
C  10.0
C  10.0
C  10.0
C  10.0
C
```

```
C  THE RESULTING OUTPUT FILE 'STATEQ.OUT' IS:
C
C                   INPUT PARAMETERS:
C
C     INPUT DATA SOURCE FILE: STATEQ.TST
C     THE NUMBER OF STATES IS: N =  2
C     THE NUMBER OF SYSTEM INPUTS IS: M = 1
C     THE NUMBER OF SYSTEM OUTPUTS IS: Q = 1
C     THE VALUE OF nstop IS: nstop =  3
C     THE VALUE FOR xsorce IS: F
C
C     THE MATRIX A(i,j) IS:
C
C     .0000E+00  -.1000E+01
C     .1000E+01   .0000E+00
C
C     THE MATRIX B(i,j) IS:
C
C     .1000E+01
C     .0000E+00
C
C     THE MATRIX C(i,j) IS:
C
C     .1000E+01  -.1000E+01
C
C     THE MATRIX D(i,j) IS:
C
C     .1000E+01
C
C     THE INITIAL CONDITION OF THE STATE VECTOR IS:
C
C       v1 =  .500000E+01
C       v2 = -.500000E+01
C
C
C                   OUTPUT DATA:
C
C    FOR ns =  0 THE STATE OF THE SYSTEM IS:
C     THE VECTOR x is:
C       x1 =  .100000E+02
C     THE VECTOR v is:
C       v1 =  .500000E+01
C       v2 = -.500000E+01
C     THE VECTOR y is:
C       y1 =  .200000E+02
C
C    FOR ns =  1 THE STATE OF THE SYSTEM IS:
C     THE VECTOR x is:
C       x1 =  .100000E+02
C     THE VECTOR v is:
C       v1 =  .150000E+02
C       v2 =  .500000E+01
```

```
C     THE VECTOR y is:
C       y1 =  .200000E+02
C
C    FOR ns =  2 THE STATE OF THE SYSTEM IS:
C     THE VECTOR x is:
C       x1 =  .100000E+02
C     THE VECTOR v is:
C       v1 =  .500000E+01
C       v2 =  .150000E+02
C     THE VECTOR y is:
C       y1 =  .000000E+00
C
C    FOR ns =  3 THE STATE OF THE SYSTEM IS:
C     THE VECTOR x is:
C       x1 =  .100000E+02
C     THE VECTOR v is:
C       v1 = -.500000E+01
C       v2 =  .500000E+01
C     THE VECTOR y is:
C       y1 =  .000000E+00
C
C
C   FOR ILLUSTRATIVE PURPOSES THE SAME INPUT SEQUENCE COULD HAVE BEEN
C   GENERATED BY CHANGING xsorce TO 'S' AND USING SUBROUTINE xgen
C   TO GENERATE THE VALUES FOR xs(1,ns).  THE APPROPRIATE FORTRAN
C   STATEMENTS ARE WRITTEN INTO xgen BUT 'COMMENTED OUT' FOR THIS
C   EXAMPLE.
C
C
C****************************  MAIN PROGRAM  ****************************


      real A(10,10), B(10,4), C(4,10), D(4,4), jj
      real ys(4,0:99), vs(10,0:100), xs(4,0:99), v(10)
      integer Q
      character*1 mode, xsorce, infile*12









C  PROMPT USER FOR MODE: BATCH OR TEST.

      write(*,1115)
      read(*,1117) mode
      if((mode.eq.'Y').or.(mode.eq.'y')) then
     mode = 'Y'
```

```
      write(*,1118)
      read(*,1119) infile
       else
      infile = 'STATEQ.IN'
       endif

C  UNIT=1 DEFINED AS INPUT FILE.  UNITS=2,3 DEFINED AS OUTPUT FILES.

       open(unit=1,file=infile,status='old',iostat=ierr,err=999)
       open(unit=2,file='STATEQ.OUT')
       open(unit=3,file='STATEQ.DAT')

C  READ INPUT PARAMETERS AND PERFORM ERROR CHECKS.

       read(1,1001) N, M, Q
       read(1,1002) nstop, xsorce

       if((N.lt.0).or.(N.gt.10)) then
      write(*,1124) 'N', N
      stop 'The allowed values for ''N'' are:  0 <= N <= 10.'
       endif

       if((M.lt.0).or.(M.gt.4)) then
      write(*,1124) 'M', M
      stop 'The allowed values for ''M'' are:  0 <= M <= 4.'
       endif

       if((Q.lt.0).or.(Q.gt.4)) then
      write(*,1124) 'Q', Q
      stop 'The allowed values for ''Q'' are:  0 <= Q <= 4.'
       endif

       if((nstop.lt.0).or.(nstop.gt.99)) then
      write(*,1127) nstop
      stop 'The allowed values for nstop are: 0 < nstop <= 99.'
       endif

       if((xsorce.eq.'F').or.(xsorce.eq.'f')) then
      xsorce = 'F'
       elseif((xsorce.eq.'S').or.(xsorce.eq.'s')) then
      xsorce = 'S'
       else
      write(*,1128)  xsorce
      stop 'The allowed values for ''xsorce'' are: ''F'' or ''S''.'
       endif

C  FOR TEST MODE ECHO INPUT PARAMETERS ONTO MONITOR (UNIT = *).

       if(mode.eq.'Y') then
         write(*,1006)
         write(*,1007) infile
         write(*,1008) N
         write(*,1009) M
         write(*,1010) Q
         write(*,1011) nstop
      write(*,1012) xsorce
      if(N.eq.0) write(*,1131)
       endif

C  WRITE INPUT PARAMETERS N,M,Q,nstop,xsorce INTO FILE: STATEQ.OUT.

       write(2,1006)
       write(2,1007) infile
       write(2,1008) N
       write(2,1009) M
       write(2,1010) Q
       write(2,1011) nstop
       write(2,1012) xsorce

C  READ SYSTEM MATRICES AND WRITE THEM INTO FILE: STATEQ.OUT.

       if(N.eq.0) then
      write(2,1131)

       else
         write(2,1110) 'A(i,j)'
      do 30 i=1, N
        read(1,1003) (A(i,j),j=1,N)
           write(2,1005) (A(i,j),j=1,N)
30       continue

         write(2,1110) 'B(i,j)'
      do 40 i=1, N
           read(1,1004) (B(i,j),j=1,M)
           write(2,1005) (B(i,j),j=1,M)
40       continue

      write(2,1110) 'C(i,j)'
      do 50 i=1, Q
        read(1,1003) (C(i,j),j=1,N)
        write(2,1005) (C(i,j),j=1,N)
50       continue
       endif


       write(2,1110) 'D(i,j)'
       do 54 i=1, Q
      read(1,1004) (D(i,j),j=1,M)
      write(2,1005) (D(i,j),j=1,M)
54     continue

C  READ THE INITIAL CONDITION VECTOR v() AND WRITE THE VALUES
C  INTO FILE: STATEQ.OUT.
```

```
      if(N.gt.0) then
     read(1,1003) (v(i),i=1,N)
     write(2,1123)
     do 64 i=1, N
       write(2,1133) 'v', i, v(i)
       vs(i,0) = v(i)
64      continue

C  FOR TEST MODE WRITE THE VECTOR v() ONTO THE MONITOR.

     if(mode.eq.'Y') then
       write(*,*)
       pause '---------->  Type <CR> to continue.  <----------'
       write(*,1123)
       do 65 i=1, N
         write(*,1133) 'v', i, v(i)
65        continue
     endif
      endif

C  FOR xsorce = 'F' READ THE ARRAY xs(i,j) FROM THE INPUT FILE.
C  FOR xsorce = 'S' CALL 'xgen' TO GENERATE THE ARRAY xs(i,j).

      if(xsorce.eq.'F') then
     do 70 j=0, nstop
       read(1,1004) (xs(i,j),i=1,M)
70      continue
      else
     call xgen(xs,M,nstop)
      endif

C  COMPUTE ITERATIVE SOLUTION TO THE SYSTEM OF EQUATIONS.

      call itrate(N,M,Q,nstop,A,B,C,D,xs,vs,ys)

C  FOR EACH VALUE OF ns WRITE THE VALUES OF xs(), vs(), ys() TO THE
C  OUTPUT FILE: STATEQ.OUT.

      write(2,1129)

      do 100 ns=0, nstop
     write(2,1130) ns
     write(2,1132) 'x'
     do 101 i=1, M
       write(2,1133) 'x', i, xs(i,ns)

101     continue

        if(N.gt.0) then
          write(2,1132) 'v'
       do 102 i=1, N
         write(2,1133) 'v', i, vs(i,ns)
```

```
102         continue
          endif

          write(2,1132) 'y'
       do 103 i=1, Q
         write(2,1133) 'y', i, ys(i,ns)
103       continue
100     continue

C  WRITE THE SEQUENCES xs, vs, ys INTO THE FILE: STATEQ.DAT.

        numplts = N + M + Q
        if(numplts.gt.9) then
       numplts = 9
       N = 9 - M - Q
        endif

        write(3,2000) numplts

        do 55 i=1, M
       write(3,2001) nstop+1
       write(3,2002) 'INPUT SEQUENCE x', char(48+i), ' (ns)'
       write(3,*) 'SAMPLE # (ns)'
       write(3,2003) 'x', char(48+i), ' (ns)'
       do 56 ns=0, nstop
         jj = ns
         write(3,2010) jj, xs(i,ns)
56        continue
55      continue

        do 57 i=1, N
       write(3,2001) nstop+1
       write(3,2002) 'STATE SEQUENCE v', char(48+i), ' (ns)'
       write(3,*) 'SAMPLE # (ns)'
       write(3,2003) 'v', char(48+i), ' (ns)'
       do 58 ns=0, nstop
         jj = ns
         write(3,2010) jj, vs(i,ns)
58        continue
57      continue

        do 59 i=1, Q
       write(3,2001) nstop+1
       write(3,2002) 'OUTPUT SEQUENCE y', char(48+i), ' (ns)'
       write(3,*) 'SAMPLE # (ns)'
       write(3,2003) 'y', char(48+i), ' (ns)'
       do 60 ns=0, nstop
         jj = ns
         write(3,2010) jj, ys(i,ns)
60        continue
59      continue
```

```
      write(*,1121)
999   close(unit=1)
      close(unit=2)
      close(unit=3)

      if(ierr.gt.0) then
     write(*,1116) ierr
      endif

C********  INPUT FORMAT  ********

1001  format(i2,t11,i1,t21,i1)
1002  format(i2,t11,a1)
1003  format(6(f10.0))
1004  format(4(f10.0))

C*********************************
1005  format(6(2X,e10.4))
1006  format(t16,//////,' INPUT PARAMETERS:',//)
1007  format(t4,'INPUT DATA SOURCE FILE: ',a12)
1008  format(t4,'THE NUMBER OF STATES IS: N = ',i2)
1009  format(t4,'THE NUMBER OF SYSTEM INPUTS IS: M = ',i1)
1010  format(t4,'THE NUMBER OF SYSTEM OUTPUTS IS: Q = ',i1)
1011  format(t4,'THE VALUE OF nstop IS: nstop = ',i2)
1012  format(t4,'THE VALUE FOR xsorce IS: ',a1,/)
1110  format(///,t4,'THE MATRIX ',a6,' IS:',/)
1115 0format(1x,'DO YOU WISH TO RUN THIS PROGRAM IN TEST',
     1' MODE ?    (Y/N) <CR> : ',\,)
1116 0format(///,1x,'ERROR OPENING INPUT FILE, PROGRAM TERMINATED.',
     1//,1x,'ERROR CODE:',i4,/////)
1117  format(a1)
1118 0format(/////,1x,'TYPE THE NAME OF YOUR DATA FILE FOLLOWED',
     1' BY <CR>.',/,' IF YOU DESIRE TO MAKE A TEST RUN USING THE',
     2' SAMPLE DATA ALREADY STORED',/,' IN THE FILE: STATEQ.TST',
     3 '        TYPE: STATEQ.TST <CR>',/,' FILENAME: ',\,)
1119  format(a12)
1121 0format(//,' TABULAR OUTPUT DATA IS STORED IN FILE: STATEQ.OUT.',
     1/,' PLOTTING DATA IS STORED IN FILE: STATEQ.DAT.')
1123  format(//,t4,'THE INITIAL CONDITION OF THE STATE VECTOR IS:',/,)
1124 0format(//,' The value of ',a1,' is ',i2,'.  This value is not
     1 allowed.')
1127 0format(//,' The value of nstop is ',i2,'.  This value is not
     1 allowed.')
1128 0format(//,' The value of xsorce is ',a1,'.  This value is not
     1 allowed.')
1129  format(//,t16,' OUTPUT DATA:')
1130  format(//,t2,'FOR ns = ',i2,' THE STATE OF THE SYSTEM IS:')
1131  format(/,' THIS SYSTEM IS NON-RECURSIVE !!!')
1132  format(t4,'THE VECTOR ',a1,' is:')
1133  format(t6,a1,i1,' = ',e12.6)
2000  format(i1)
2001  format(i3)
```

```
2002  format(a17,a1,a5)
2003  format(a1,a1,a5)
2010  format(e12.6,2x,e12.6)

      end


C                         SUBROUTINE: itrate


C  PURPOSE:  THIS SUBROUTINE COMPUTES THE ITERATIVE SOLUTION TO
C            THE DISCRETE, STATE MATRIX SYSTEM OF EQUATIONS.
C            THE INPUTS TO THIS SUBROUTINE ARE THE DIMENSIONS
C            N,M,Q, THE PARAMETER nstop, AND THE 2-DIMENSIONAL
C            ARRAYS A, B, C, D, xs, AND vs.  FOR EACH VALUE OF ns
C            FROM ns = 0 TO ns = nstop THE STATE OF THE SYSTEM vs
C            IS COMPUTED AS IS THE CORRESPONDING OUTPUT ARRAY ys.


      subroutine itrate(N,M,Q,nstop,A,B,C,D,xs,vs,ys)
      integer Q
      real A(10,10), B(10,4), C(4,10), D(4,4)
      real xs(4,0:99), vs(10,0:100), ys(4,0:99)

      do 1 ns=0, nstop

C  FOR ns = 0 TO nstop: COMPUTE THE SOLUTION
C  TO THE EQUATION:  vs(ns+1) = A*vs(ns) + B*xs(ns).

     do 2  i=1, N
       xi = 0.0
       vs(i,ns+1) = 0.0
       do 3 k=1, M
         xi = xi + B(i,k)*xs(k,ns)
3          continue

       do 4 j=1, N
         vs(i,ns+1) = vs(i,ns+1) + A(i,j)*vs(j,ns)

4          continue
       vs(i,ns+1) = vs(i,ns+1) + xi
2         continue



C  COMPUTE THE SOLUTION TO THE EQUATION: ys(ns) = C*vs(ns) + D*xs(ns).

     do 5 l=1, Q
       ys(l,ns) = 0.0
       xi = 0.0
       do 6  k=1, M
```

```
         xi = xi + D(l,k)*xs(k,ns)
6         continue
       do 7 j=1, N
         ys(l,ns) = ys(l,ns) + C(l,j)*vs(j,ns)
7         continue
       ys(l,ns) = ys(l,ns) + xi
5       continue

1     continue

      return
      end


C                         SUBROUTINE: xgen


C  PURPOSE:  THIS SUBROUTINE ALLOWS THE USER TO GENERATE VALUES FOR
C            THE 2-DIMENSIONAL ARRAY xs(i,j).  IF xsorce = 'S' THE
C            MAIN PROGRAM WILL CALL THIS SUBROUTINE. IF xsorce = 'F'
C            THIS SUBROUTINE WILL NOT BE CALLED BY THE MAIN PROGRAM.


      subroutine xgen(xs,M,nstop)
      real xs(4,0:99)

      pi = 4.0*atan(1.0)

C*************************************************************************

C  DEVELOP THE ALGORITHM FOR GENERATING VALUES OF xs(i,j) IN THIS
C  SPACE. THE STATEMENTS TYPED IN MUST FOLLOW STANDARD FORTRAN 77
C  RULES AND MAY USE FORTRAN 77 INTRINSIC FUNCTIONS SUCH AS: SIN(),
C  COS(),...  NOTE THAT THE ROW INDEX i DENOTES THE INPUT SEQUENCE
C  NUMBER 1 <= i <= M, AND THE INDEX j DENOTES THE VALUE FOR ns
C  0 <= ns <= nstop.  AN EXAMPLE OF AN ALGORITHM GENERATING VALUES
C  FOR THE SEQUENCE xs(1,ns) IS:
C
C
C***  EXAMPLE  ***
C       do 88 ns=0, nstop
C         xs(1,ns) = 10.0
C 88    continue

C*************************************************************************


      return
      end
```

```
C                               PLOTDAT.FOR       VERSION: 2/03/88
C
C
C  PURPOSE:  THIS PROGRAM IS DESIGNED TO CREATE UP TO NINE TWO-
C            DIMENSIONAL (2-D) PLOTS BY READING DATA FROM AN INPUT
C            FILE AND PLOTTING THE DATA ON THE MONITOR SCREEN.  THE
C            PROGRAM CONSISTS OF A MAIN PROGRAM AND THE SUBROUTINES
C            scale AND gridd. THE MAIN PROGRAM READS THE DATA FROM THE
C            INPUT FILE, AND CREATES THE PLOT(S) BY MAKING CALLS TO
C            THE GRAPHMATICS GRAPHICS LIBRARY (NOTE 1).  SUBROUTINE
C            scale IS CALLED BY THE MAIN PROGRAM TO SCALE THE DATA SO
C            AS TO OPTIMALLY FILL THE SCREEN WITH EACH PLOT. SUBROUTINE
C            gridd IS CALLED BY THE MAIN PROGRAM IF THE USER ELECTS TO
C            HAVE A GRID OVERLAY THE PLOT.  THE USER HAS THE OPTION OF
C            PERFORMING A TRIAL RUN BY PLOTTING THE DATA PRESTORED IN
C            THE FILE 'PLOTDAT.TST'. IT IS RECOMMENDED THAT FIRST-TIME
C            USERS MAKE A TRIAL RUN BY SPECIFYING THE INPUT FILE
C            'PLOTDAT.TST' WHEN PROMPTED FOR THE NAME OF THE INPUT
C            FILE. THE SPECIFIC HARDWARE REQUIREMENTS NECESSARY TO RUN
C            THE PROGRAM ARE OUTLINED BELOW.  ADDITIONALLY, THE INPUT
C            FORMAT REQUIREMENTS ARE PRESENTED AND A SAMPLE INPUT FILE
C            LISTING IS INCLUDED.
C
C  NOTE 1.   COPYRIGHT 1984, MICROCOMPATIBLES INC., SILVER SPRINGS, MD.
C
C
C**********************  HARDWARE REQUIREMENTS  ************************

C
C
C  THIS PROGRAM WAS WRITTEN FOR PERSONAL COMPUTERS OUTFITTED WITH A
C  COLOR GRAPHICS ADAPTER (CGA) CARD.  THE MONITOR SCREEN IS ASSUMED
C  TO BE 640 X 200 PIXELS.  FOR HARDCOPY PRINTOUTS OF THE PLOTS THE
C  PROGRAM WILL DRIVE A DOT MATRIX PRINTER IF INSTALLED.  FOR
C  COMPUTERS OUTFITTED WITH OTHER THAN A CGA CARD THE PROGRAM MAY NOT
C  OPERATE.  FOR MACHINES THAT HAVE AN EGA CARD THE PROGRAMS WILL NOT
C  PRODUCE A HARDCOPY PRINTOUT OF THE PLOTS.  TO OVERCOME THIS
C  LIMITATION, USERS SHOULD TRY EXECUTING THE 'PRINT SCREEN' COMMAND.
C
C
C******************************  INPUT  ********************************

C
C
C  UPON EXECUTION OF THE PROGRAM, THE USER IS PROMPTED FOR THE NAME
C  OF THE INPUT FILE. THE INPUT FORMAT STATEMENTS OCCUR IN THE MAIN
C  PROGRAM FOLLOWING THE CAPTION:  ******** INPUT FORMAT ********.
C  THE FORM OF THE INPUT DATA FILE IS:
C
```

```
C  LINE #         ENTRIES          FORMAT           RESTRICTIONS
C  ------         -------          ------           ------------
C    1            numplts            i1          1 <= numplts <= 9
C    2            numpts             i3          2 <= numpts <= 999
C    3            title              a40
C    4            xlabl              a14
C    5            ylabl              a14
C    6 ...        x(), y()       f12.0,2X,f12.0        NOTE 2
C  NOTE 1
C
C  WHERE:
C
C  numplts = AN INTEGER VALUE THAT SPECIFIES THE NUMBER OF PLOTS TO
C            BE CREATED BY THE PROGRAM. FOR EACH PLOT 1, ..., numplts
C            THE INPUT DATA SPECIFIED BELOW MUST OCCUR IN THE INPUT
C            FILE.
C
C  numpts =  AN INTEGER VALUE THAT SPECIFIES THE NUMBER OF POINTS TO
C            BE READ FROM THE INPUT FILE, FOR A GIVEN PLOT, AND
C            PLOTTED.
C
C  title  =  A CHARACTER STRING CONSISTING OF UP TO 40 CHARACTERS.
C            THIS STRING IS PLACED ABOVE THE PLOT.
C
C  xlabl  =  A CHARACTER STRING CONSISTING OF UP TO 14 CHARACTERS.
C            THIS STRING IS PLACED BENEATH THE X-AXIS.
C
C  ylabl  =  A CHARACTER STRING CONSISTING OF UP TO 14 CHARACTERS.
C            THIS STRING IS PLACED ADJACENT TO THE Y-AXIS.
C
C  x() y() = LINES 6...6+numpts MUST EACH CONTAIN A PAIR OF REAL
C            NUMBERS THAT COMPRISE THE ORDINATE x() AND ABSCISSA y()
C            VALUES DEFINING A SINGLE POINT TO BE PLOTTED.
C
C  NOTE 1.   THE DATA REQUIRED FOR LINES 2 ... 6+numpts IS REPEATED
C            FOR EACH PLOT (1...numplts) TO BE CREATED BY THE PROGRAM.
C
C  NOTE 2.   THE FORMAT f12.0 USED FOR INPUT DATA PERMITS THE DECIMAL
C            POINT TO BE PLACED ANYWHERE IN THE FIELD OF 12 COLUMNS
C            AND ALSO ALLOWS THE EXPONENTIAL FORMAT TO BE USED (E.G.,
C            3146.2 = 3.1462E+03).
C
C
C*****************************  EXAMPLE  ******************************

C
C
C  PRINTED BELOW IS A LISTING OF ONE OF THE TWO EXAMPLES INCLUDED IN
C  THE INPUT FILE 'PLOTDAT.TST'.  THE ENTRIES ON EACH LINE SHOULD BE
C  COMPARED TO THE FORMAT REQUIREMENTS LISTED ABOVE.
C
C
```

```
C  1
C  005
C  title (UP TO 40 CHARS) IS PRINTED HERE
C  xlabl HERE
C  ylabl HERE
C  0.0          0.0
C  1.0          1.0
C  2.0          2.0
C  3.0          3.0
C  4.0          4.0
C
C
C***************************  MAIN PROGRAM  ***************************


$STORAGE:2

      character copy*1, yscal*3, xlabl*14, ylabl*14, scal*6
      character title*40, infile*12, plot*1, xx*1, yy*1, grid*1
      real x(999), y(999), dum(4,999)
      integer tmode,gmode

C  DEFINE SCREEN SIZE:  640 x 200 PIXELS

      gmode = 6
      tmode = 2

C  DEFAULT PLOT TYPE: SOLID BLACK LINE, POINTS CONNECTED BY A LINE.

      ndots = 0
      icolor = 3
      klrsym = 3

C  ENABLE MINOR TIC MARKS ON AXES.
C  ENABLE MAJOR TIC MARKS AND LABEL TO AN ACCURACY OF 0.01.

      minorx = 1
      minory = 1
      label = -1
      ndec = 2

C  ENABLE  PLOT AUTO-SCALING.
C  y/x RATIO OF PLOT = 1.0.
C  PLOT ASPECT RATIO = 1.2.

      io = 0
      yx = 1.0
      aspct = 1.2
```

```
C  DEFINE CHARACTER STRING CONSTANTS.

      scal = 'x 10**'
      xx = 'x'
      yy = 'y'

C  CLEAR SCREEN

      call qclear(0,7)

C  OPEN INPUT FILE AND READ THE VALUE numplts.

      write(*,111)
      read(*,109) infile

      open(unit=1,file=infile,status='old',iostat=ierr,err=999)
      read(1,100) numplts

C  PROMPT USER FOR DESIRED OPTIONS.

      do 5 i=1, numplts
     data x/999*0.0/, y/999*0.0/
     write(*,107)
     read(*,101) copy
     write(*,112)
     read(*,101) grid
     write(*,108)
     read(*,101) plot

C  READ HEADER DATA FROM INPUT FILE.

     read(1,102) numpts
     read(1,104) title
     read(1,105) xlabl
     read(1,105) ylabl

C  FOR PLOTS OF 25 POINTS OR LESS, PLOTTING SYMBOL = '+'.

     if(numpts.le.25) then
       isymbol = 43
       itype = 0
     else
       isymbol = -2
       itype = 1
     endif

C  SCALE THE INPUT DATA.

     xmin =  3.0e+38
     xmax = -3.0e+38
     ymin =  3.0e+38
     ymax = -3.0e+38
```

```
      do 10 k=1, numpts
        read(1,106) x(k), y(k)
        xmax = max(x(k),xmax)
        xmin = min(x(k),xmin)
        ymax = max(y(k),ymax)
        ymin = min(y(k),ymin)
10         continue

      if(xmax.eq.xmin) then
        write(*,*) 'Execution halted.'
        write(*,*) 'The increment of the ordinate values = 0.0.'
        stop 'Check the ordinate values in the input file.'
      endif

      if((ymax.eq.0.0).and.(ymin.eq.0.0)) then
        write(*,*) 'For the current plot, the maximum and minimum'
        write(*,*) 'abscissa values are:  ymax = ymin = 0.0.'
        write(*,*) 'Data ignored ... computing next plot.'
        goto 5
      endif

      call scale(xmin,xmax,ixscal,xx)
      call scale(ymin,ymax,iyscal,yy)
      scalx = real(ixscal)
      scaly = real(iyscal)

      xmajor = abs(xmax-xmin)/5.0 - 0.000001
      ymajor = abs(ymax-ymin)/5.0 - 0.000001

      do 11 k=1, numpts
        x(k) = x(k)/(10.0**ixscal)
        y(k) = y(k)/(10.0**iyscal)
11         continue

C  BEGIN GRAPHICS SECTION.

      call qsmode(gmode)
      call qplot(160,600,30,180,xmin,xmax,ymin,ymax,xmin,ymin,io,yx,
      &              aspct)
      call qsetup(ndots,icolor,isymbol,klrsym)
      call qptxt(40,title,icolor,29,24)

      call qxaxis(xmin,xmax,xmajor,minorx,label,ndec)
      call qptxt(14,xlabl,icolor,40,0)
      call qptxt(6,scal,icolor,68,1)
      call qqnput(576,8,scalx,0)

      call qyaxis(ymin,ymax,ymajor,minory,label,ndec)
      call qptxt(14,ylabl,icolor,0,12)
      call qptxt(6,scal,icolor,0,22)
      call qqnput(28,176,scaly,0)
```

```
C  OVERLAY GRID ONTO THE PLOT IF SPECIFIED.

      if((grid.eq.'Y').or.(grid.eq.'y')) then
        call gridd()
      endif

C  PLOT THE POINTS OF THE ARRAYS.

      call qtabl(itype,numpts,x,y)

C  PRINT HARDCOPY IF SPECIFIED.

      if((copy.eq.'y').or.(copy.eq.'Y')) then
        call qpscrn
      endif

      call qinkey(iextend,key)
      call qsmode(tmode)

 5     continue

       write(*,*) 'Plotting completed, returning to DOS.'

999    close(unit=1)
       if(ierr.gt.0) then
      write(*,110) infile, ierr
       endif

C******** INPUT FORMAT ********

100    format(i1)
102    format(i3)
104    format(a40)
105    format(a14)
106    format(f12.0,2x,f12.0)

C******************************

101    format(a1)
107   0format(1x,'Do you want a hardcopy of the plot to be',
      1' generated next ?  Y/N <CR>',\)
108   0format(1x,'To begin the plotting or to exit from the',
      1' plotting enter <CR>.',\)
109    format(a12)
110    format(1x,'Error opening file: ',a12,'  Error code = ',i4)
111   0format(/////,1x,'TYPE THE NAME OF YOUR DATA FILE FOLLOWED',
      1' BY <CR>.',/,' IF YOU DESIRE TO MAKE A TEST RUN USING THE',
      2' SAMPLE DATA ALREADY STORED',/,' IN THE FILE: PLOTDAT.TST',
      3'      TYPE: PLOTDAT.TST <CR>',/,' FILENAME:',\,)
112    format(1x,'Do you want a grid to overlay the plot ?  Y/N <CR>',\)

       end
```

```
C                             SUBROUTINE: scale

C  PURPOSE:  THIS SUBROUTINE FINDS THE LARGEST INTEGER POWER OF TEN
C            OCCURRING IN EITHER valmin OR valmax.  THE RESULTING
C            EXPONENT IS RETURNED TO THE CALLING PROGRAM IN iscal.
C            THE SUBROUTINE ALSO SCALES valmin AND valmax BEFORE
C            RETURNING THESE VALUES TO THE MAIN PROGRAM.


      subroutine scale(valmin,valmax,iscal,c)
      character c*1

      iscal = 0
      arg1 = 0.0
      arg2 = 0.0

C  FIND THE LARGEST INTEGER POWER OF 10 IN THE SEQUENCE.

      if(valmax.ne.0.0) then
     arg1 = log10(abs(valmax))
      endif

      if(valmin.ne.0.0) then
     arg2 = log10(abs(valmin))
      endif

      iscal = int(max(arg1,arg2))

C  SCALE THE MAXIMUM AND MINIMUM VALUES OF THE SEQUENCE.

      valmin = valmin/(10.0**iscal)
      valmax = valmax/(10.0**iscal)

C  CREATE A BUFFER SPACE FOR THE ABSCISSA VALUES OF THE PLOT.

      if(c.eq.'y') then

     if((valmin.lt.0.0).and.(valmax.lt.0.0)) valmax = 0.0
     if((valmin.gt.0.0).and.(valmax.gt.0.0)) valmin = 0.0
     tempmin = anint(-1.0+valmin)
     tempmax = anint(1.0+valmax)

     if(valmin.ne.0.0) then
 2        if(abs(valmin) - 0.1*abs(tempmin)) 3,4,4
 3          tempmin = .1*tempmin
        goto 2
 4          valmin = valmin + .1*tempmin
     endif

     if(valmax.ne.0.0) then
 5        if(abs(valmax) - 0.1*abs(tempmax)) 6,7,7
 6          tempmax = .1*tempmax
         go to 5
 7          valmax = valmax + .1*tempmax
       endif
     endif

      return
      end



C                          SUBROUTINE: gridd


C  PURPOSE:  THIS SUBROUTINE IS CALLED BY THE MAIN PROGRAM TO CREATE
C            A GRID OVERLAY CONSISTING OF BOTH HORIZONTAL AND VERTICAL
C            DASHED LINES EXTENDING FROM THE MAJOR TIC MARKS OF THE
C            AXIS.


      subroutine gridd()

C  CREATE HORIZONTAL DASHED LINES AT THE MAJOR TIC MARKS OF THE PLOT.

      do 500 i=1, 5
     ii = 29 + 30*i
     call qdash(5,160,ii,600,ii,3)
500   continue

C  CREATE VERTICAL DASHED LINES AT THE MAJOR TIC MARKS OF THE PLOT.

      do 501 j=1, 5
     jj = 159 + 88*j
     call qdash(7,jj,30,jj,179,3)
501   continue

      return
      end
```

LIST OF REFERENCES

1. Strum, R. D. and Kirk, D. E., First Principles of Discrete Systems and Digital Signal Processing, Addison-Wesley Publishing Co., 1988.

2. Etter, D. M., Problem Solving with Structured Fortran 77, Benjamin/Cummings Publishing Co., Inc., 1984.

3. Gerald, C. F. and Wheatley, P. O., Applied Numerical Analysis, 3d ed., p.20, Addison-Wesley Publishing Co., 1984.

Bibliography

Brigham, E. O., The Fast Fourier Transform, Prentice-Hall Inc., 1974.

Dudgeon, D. E. and Mersereau, R. M., Multidimensional Digital Signal Processing, Prentice-Hall Inc., 1984.

Gonzalez, R. C. and Wintz, P., Digital Image Processing, Addison-Wesley Publishing Co., 1977.

Kirk, D. E., Various Fortran algorithms for the solution of Digital Signal Processing problems., 1987.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Professor D. E. Kirk, Code 62Ki Naval Postgraduate School Monterey, California 93943-5000	5
4.	Professor R. D. Strum, Code 62St Naval Postgraduate School Monterey, California 93943-5000	3
5.	Professor C. W. Therrien, Code 62Ti Naval Postgraduate School Monterey, California 93943-5000	1
6.	Lt. J. V. England, Code 62Eg Naval Postgraduate School Monterey, California 93943-5000	1
7.	Strategic Systems Project Office Atten: Fred Wimberly, SP 27331 617 21st. Street South Arlington, Virginia 22202	1
8.	Professor R. Santoro Department of Electrical Engineering U. S. Naval Academy Annapolis, Maryland 21412	1
9.	Lt. F. E. Hudik 17103 Lake Point Drive S.E. Yelm, Washington 98597	1

Thesis
H8475 Hudik
c.1 A computer program package for introductory one-dimensional digital signal processing applications.

8 FEB 91 37019

Thesis
H8475 Hudik
c.1 A computer program package for introductory one-dimensional digital signal processing applications.